

```
SSSSSSSSSSSSSS 000000000 000 RRRRRRRRRRRR TTTTTTTTTTTTTTTT 333333333 222222222
SSSSSSSSSSSSSS 000000000 000 RRRRRRRRRRRR TTTTTTTTTTTTTTTT 333333333 222222222
SSSSSSSSSSSSSS 000000000 000 RRRRRRRRRRRR TTTTTTTTTTTTTTTT 333333333 222222222
SSS 000 000 RRR RRR 333 333 222 222
SSS 000 000 RRR RRR 333 333 222 222
SSS 000 000 RRR RRR 333 333 222 222
SSS 000 000 RRR RRR 333 333 222 222
SSS 000 000 RRR RRR 333 333 222 222
SSSSSSSSSS 000 000 RRRRRRRRRRRR TTT TTT 333 333 222 222
SSSSSSSSSS 000 000 RRRRRRRRRRRR TTT TTT 333 333 222 222
SSSSSSSSSS 000 000 RRRRRRRRRRRR TTT TTT 333 333 222 222
SSS 000 000 RRR RRR 333 333 222 222
SSS 000 000 RRR RRR 333 333 222 222
SSS 000 000 RRR RRR 333 333 222 222
SSS 000 000 RRR RRR 333 333 222 222
SSS 000 000 RRR RRR 333 333 222 222
SSSSSSSSSSSS 000000000 000 RRR RRR TTT 333333333 2222222222222222
SSSSSSSSSSSS 000000000 000 RRR RRR TTT 333333333 2222222222222222
SSSSSSSSSSSS 000000000 000 RRR RRR TTT 333333333 2222222222222222
```

SRT
V04

.....

[illegible]


```
1 0001 0 MODULE SRTSPC ( IDENT = 'V04-000'      ! File: SRTSPCBLI.B32 Edit: VLB3042
2 L 0002 0      %REQUIRE('SRC$:DEFS0') ) =
3 0008 1 BEGIN
4 0009 1 -----
5 0010 1
6 0011 1 *****
7 0012 1 *
8 0013 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0014 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0015 1 *  ALL RIGHTS RESERVED.
11 0016 1 *
12 0017 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0018 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0019 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0020 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0021 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0022 1 *  TRANSFERRED.
18 0023 1 *
19 0024 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0025 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0026 1 *  CORPORATION.
22 0027 1 *
23 0028 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0029 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0030 1 *
26 0031 1 *
27 0032 1 *****
28 0033 1 -----
29 0034 1
30 0035 1
31 0036 1 ++
32 0037 1 Facility:
33 0038 1
34 0039 1     PDP-11 SORT/MERGE V3.0
35 0040 1     VAX-11 SORT/MERGE V3.0
36 0041 1
37 0042 1 Abstract:
38 0043 1
39 0044 1
40 0045 1 Environment:
41 0046 1
42 0047 1
43 0048 1 Author:
44 0049 1
45 0050 1     Victor L Bennison,      Creation Date:      8-Mar-82
46 0051 1
47 0052 1 MODIFIED BY :
48 0053 1     X30-001 Peter D Gilbert      Aug 1982
49 0054 1     Pervasive modifications.
50 0055 1     Pass text as a vector of 2 or 3 items.
51 0056 1     Move expansion of keywords into this module.
52 0057 1     General code cleanups.
53 0058 1     Added ERR_ and WRN_ macros.
54 0059 1     Corrected the EBC TAB table.
55 0060 1     Used TYP_K_xxx values to identify the sort process.
56 0061 1     Deferred determining whether sort or merge until run-time.
57 0062 1     Used DT_xxx values to identify datatypes.
```



```
58 0063 1 T03-016 Improve allocation of the work area. Change PARSE_TEST for
59 0064 1 precedence of AND/OR, and nested parens. PDG 13-Dec-1982
60 0065 1 T03-017 Complete implementation of /WORK_FILES; store a counted list of
61 0066 1 CFT table indices for the work file names. PDG 26-Dec-1982
62 0067 1 T03-018 Fix a reference in PARSE_KEY which caused ASCENDING and
63 0068 1 DESCENDING to not be parsed. Also, fixed a reference in
64 0069 1 INIT_CS_TAB. PDG 28-Dec-1982
65 0070 1 T03-019 Fix off-by-one problems with ST indices.
66 0071 1 Add ZONED.
67 0072 1 Support conversions of decimal constants.
68 0073 1 Fix off-by-one problem with MOD:('C'<'R').
69 0074 1 PDG 29-Dec-1982
70 0075 1 T03-020 Fix bug in ignoring a range of characters. PDG 3-Jan-1983
71 0076 1 T03-021 Store PT_ST_IDX for GC_S_EQ_D. PDG 6-Jan-1983
72 0077 1 T03-022 Fix initialization of DEC_MULTI. PDG 10-Jan-1983
73 0078 1 T03-023 Collating sequence stuff. Fix GET_STRING. PDG 26-Jan-1983
74 0079 1 T03-024 Call COLL$FOLD to fold double characters. Fix GET_CHAR_CLAUSE
75 0080 1 so that returning GC_SINGLE also sets the second char the same
76 0081 1 as the first. PDG 1-Feb-1983
77 0082 1 T03-025 Diagnose '/key=(if x then lit)'. PDG 2-Feb-1983
78 0083 1 T03-026 Change DEC_UPP so different cases are equal. PDG 11-Feb-1983
79 0084 1 T03-027 Compute KFT_NDE_SIZ values. PDG 12-Feb-1983
80 0085 1 T03-028 Set the CA_TIE_BREAK flag in SOR$$DECM. 16-Feb-1983
81 0086 1 T03-029 Change DEC_UPP so sharp s is just before s. 2-Mar-1983
82 0087 1 T03-030 Remove need for COLL$COMPRESS. PDG 5-Apr-1983
83 0088 1 T03-031 Change for tie-breaking, so that [NO]TIE_BREAK turns on [off]
84 0089 1 both the second pass and third pass comparisons. Note that the
85 0090 1 second pass table is specified only for DEC_MULTI; by default,
86 0091 1 the second pass table is all zero (this is optimized away by
87 0092 1 COLL$RESULT). PDG 12-Apr-1983
88 0093 1 T03-032 Add some error reporting in PARSE_COLL. PDG 15-Apr-1983
89 0094 1 T03-033 Always reverse the sense of the tie-break. PDG 18-Apr-1983
90 0095 1 T03-034 Changes necessary for SORT-11 overlay. VLB 1-Jul-1983
91 0096 1 T03-034 Set the 'default length' for work file names. PDG 15-Aug-1983
92 0097 1 T03-035 Word align work area after allocating constants. 28-Jul-1983
93 0098 1 T03-036 Make CFT_CON_ADR relative. PDG 2-Aug-1983
94 0099 1 T03-037 Changes to merge Sort-32 and Sort-11 sources. PDG 31-Jan-1984
95 0100 1 T03-038 Change CONV_CONSTANTS to fix /WORK_FILES. PDG 16-Feb-1984
96 0101 1 T03-039 Set [CA_COM_COL_SEQ_] for SCRT-11 statistics, fix process
97 0102 1 override. VLB 22-FEB-1984
98 0103 1 T03-040 Fix the error handling for SORT-11. VLB 23-FEB-1984
99 0104 1 T03-041 Use TRUE/FALSE instead of ON/OFF when setting modified, also
100 0105 1 make some BLISS16 stuff unconditional instead. VLB 7-MAR-1984
101 0106 1 T03-042 Fix handling of WKAREA error. VLB 9-MAR-1984
102 0107 1
103 0108 1 SYNTAX:
104 0109 1 /COND = (NAME:name, TEST:test)
105 0110 1 /OMIT = (COND:cond)
106 0111 1 /INCL = (COND:cond, DATA:data, KEY:key)
107 0112 1 --
108 0113 1
109 0114 1 REQUIRE 'SRC$:DKS';
110 0412 1
111 0413 1 EXTERNAL ROUTINE
112 0414 1 P1$PASS_ONE: CA_LINKAGE,
113 0415 1 P2$PASS_TWO: CA_LINKAGE,
114 0416 1 P3$PASS_THREE: CA_LINKAGE,
```


:	115	0417	1	P4\$PASS_FOUR:	CA_LINKAGE,	
:	116	0418	1	P5\$PASS_FIVE:	CA_LINKAGE;	
:	117	0419	1			
:	118	0420	1	FORWARD ROUTINE		
:	119	0421	1	CVT_ATB,		! Convert ascii to binary
:	120	0422	1	ERR_CNTL:	CA_LINKAGE,	! Error control routine
:	121	0423	1	FIND_RPAR,		! Find the next unquoted right parenthesis
:	122	0424	1	GET_CHAR_CLAUSE,		! Get character clauses of various kinds
:	123	0425	1	GET_KYW_TYPE,		! Get keyword type
:	124	0426	1	GET_NEXT_SPEC,		! Get pointer and length of next specification
:	125	0427	1	GET_ONE_CHAR,		! Get character or double character
:	126	0428	1	GET_STRING,		! Find string and move around it
:	127	0429	1	GET_SUB_SPEC,		! Get pointer and length of sub-specification
:	128	0430	1	SEARCH_TABLE,		! Search a table for a string
:	129	0431	1	SKIP_COMMA,		! Skip commas and ignored stuff
:	130	0432	1	SKIP_IGNORED,		! Skip ignored text characters
:	131	0433	1	%IF %BLISS(BLISS32) %THEN		
:	132	0434	1	SOR\$\$DEF_FIELD:	CA_LINKAGE,	! Define a field
:	133	0435	1	%FI		
:	134	0436	1	SOR\$\$\$FPRS:	CA_LINKAGE,	! Process specification file text
:	135	0437	1	SPC_ALLOC:	CA_LINKAGE,	! Allocate storage from the work area
:	136	0438	1	SPC_HEAP:	CA_LINKAGE;	! Allocate temp storage from work area

```
: 138      0439 1  ! Define the text of the keywords
: 139      0440 1  !
: 140      0441 1  %IF %BLISS(BLISS16)
: 141      0442 1  %THEN
: 142      0443 1      PSECT
: 143      0444 1          PLIT = $$FWRD (NOWRITE,NOEXECUTE,GLOBAL);
: 144      0445 1  %FI
: 145      0446 1
: 146      0447 1  MACRO
: 147      0448 1      TXT_[A, B] = %ASCIC B %,
: 148      0449 1      OFF_[A, B] = OFF_-- %ASSIGN( OFF_--, OFF_-- + %CHARCOUNT( B ) + 1 ) %;
: 149      0450 1  COMPILETIME
: 150      0451 1      OFF_-- = 0;
: 151      0452 1  BIND
: 152      0453 1      SRTWRD = UPLIT BYTE( TXT_( KEYWORDS ) ):      VECTOR[, BYTE],
: 153      0454 1      WRDOFF = UPLIT WORD( OFF_( KEYWORDS ) ):      VECTOR[, WORD];
```



```
155 0455 1 GLOBAL ROUTINE SOR$$$FPRS (  
156 0456 1     SPEC_TEXT: REF VECTOR[2]  
157 0457 1     ): CA_LINKAGE =  
158 0458 1  
159 0459 1 ++  
160 0460 1  
161 0461 1     Functional Description:  
162 0462 1  
163 0463 1         Parses the specification file buffer  
164 0464 1  
165 0465 1     Formal Parameters:  
166 0466 1  
167 0467 1         SPEC_TEXT  Addr of length/address of buffer containing specifications.  
168 0468 1  
169 0469 1     Implicit Inputs:  
170 0470 1  
171 0471 1         None.  
172 0472 1  
173 0473 1     Implicit Outputs:  
174 0474 1  
175 0475 1         None.  
176 0476 1  
177 0477 1     Routine Value:  
178 0478 1  
179 0479 1         0 on error.  
180 0480 1         1 for sucess.  
181 0481 1  
182 0482 1     Side Effects:  
183 0483 1  
184 0484 1         None.  
185 0485 1  
186 0486 1     Notes:  
187 0487 1  
188 0488 1         When the input record LRL is known, it should be stored in:  
189 0489 1         FDT[0, FDT_FLD_SIZ] and KFT[*, KFT_NDE_SIZ] for every KFT entry  
190 0490 1         with KFT_CONSTANT = FALSE and KFT_FDT_IDX = 0.  
191 0491 1  
192 0492 1 --  
193 0493 2     BEGIN  
194 0494 2  
195 0495 2     LOCAL  
196 0496 2         TRDT: REF TRDT_TAB[RDY_MAX],           ! Hold info on record types  
197 0497 2         CDX_SYM: REF SYM_TAB[MAX_CONDX],         ! Condition "symbol" table  
198 0498 2         CDX_CNT:                                     ! Entries in CDX_SYM ???safe???  
199 0499 2         CON_SYM: REF CON_SYM_TAB[MAX_CONST],       ! Constant "symbol" table  
200 0500 2         CON_CNT:                                     ! Entries in CON_SYM ???safe???  
201 0501 2         FLD_SYM: REF SYM_TAB[MAX_FIELDS];       ! Field "symbol" table  
202 0502 2  
203 0503 2     LOCAL  
204 0504 2         START: VECTOR[3],           ! Length/address/line of start of spec buffer  
205 0505 2         WRK_END:                     ! End of work area  
206 0506 2  
207 0507 2     LABEL  
208 0508 2         normal;  
209 0509 2  
210 0510 2     ! Get the impure area  
211 0511 2     !
```

```
212 0512 2 CA_AREA_( CA );
213 0513 2
214 0514 2 output_('sor$$sfprs');
215 0515 2
216 0516 2 normal:
217 0517 2 BEGIN
218 0518 2
219 0519 2 ! Save the address at the end of the work area
220 0520 2
221 0521 2 WRK_END = .CA[CA_WRK_END];
222 0522 2
223 0523 2 ! Initialize start to length/address of buffer as passed
224 0524 2
225 0525 2 START[_LEN] = .SPEC_TEXT[_LEN];
226 0526 2 START[_PTR] = .SPEC_TEXT[_PTR];
227 0527 2 START[_LINE] = 1;
228 0528 2
229 0529 2 ! Find the start of the first specification, ignore preceding
230 0530 2 ! Comments, nulls and blanks. start will be constant for rest of parse.
231 0531 2 ! If there's no text, exit with success
232 0532 2
233 0533 2 IF SKIP_IGNORED( START[0] ) EQL FAIL THEN RETURN SUCCESS;
234 0534 2
235 0535 2 ++
236 0536 2 Seven tables are created by this routine.
237 0537 2 These tables are created/expanded in the passes listed below:
238 0538 2 CA_ST_ADR/SIZ 1
239 0539 2 CA_FDT_ADR/SIZ 2
240 0540 2 CA_TDT_ADR/SIZ 3
241 0541 2 CA_KFT_ADR/SIZ 4,5
242 0542 2 CA_RDT_ADR/SIZ 5
243 0543 2 CA_CFT_ADR/SIZ 3,4,5 (via PARSE_TEST and PARSE_KEY)
244 0544 2
245 0545 2 The CFT table may be "expanded" in any of several places, but the work
246 0546 2 area storage is not allocated until after all the other processing.
247 0547 2 The information is stored in a temporary table, described below.
248 0548 2
249 0549 2 Four temporary tables are used:
250 0550 2 FLD_SYM Symbol table for named fields
251 0551 2 CDX_SYM Symbol table for named conditions
252 0552 2 CON_SYM Symbol table for constants
253 0553 2 TRDT Temporary record definition table
254 0554 2 these are allocated at the end of the work area, and the work area
255 0555 2 restored when processing is complete.
256 0556 2 --
257 0557 2
258 0558 2 ! Initialize constant table
259 0559 2 ! Allocate space for field symbol table
260 0560 2 ! Initialize condition symbol table
261 0561 2 ! Allocate space for the temporary record definition table.
262 0562 2
263 0563 2 CON_SYM = HEAP_( %SIZE( CON_SYM_TAB[MAX_CONST] ) );
264 0564 2 CON_CNT = 0;
265 0565 2 FLD_SYM = HEAP_( %SIZE( SYM_TAB[MAX_FIELDS] ) );
266 0566 2 CDX_SYM = HEAP_( %SIZE( SYM_TAB[MAX_CONDX] ) );
267 0567 2 CDX_CNT = 0;
268 0568 2 TRDT = HEAP_( %SIZE( TRDT_TAB[TRDT_MAX] ) );
```



```
! Make an entry for the default record type.
TRDT[0, TRDT_INCLUDE] = TRUE;
TRDT[0, TRDT_CONDX ] = FALSE;
TRDT[0, TRDT_KFT_IDX] = .CALCA_KFT_SIZ];

-----
Pass 1:  parse /process, /stable, /strip,
         and /collating_sequence specifications.
         Build the PT and ST tables
-----

IF P1$PASS_ONE( START[0], CON_SYM[0, BASE_], CON_CNT ) NEQ SUCCESS
THEN
    LEAVE normal;

-----
Pass 2:  define all fields, fill the field definition
         table and the field symbol table
         Build the FDT table
-----

IF P2$PASS_TWO( START[0], FLD_SYM[0, BASE_] ) NEQ SUCCESS
THEN
    LEAVE normal;

-----
Pass 3:  parse condition specifications.
         set up test definition table,
         condition symbol table and the
         constant symbol table.
         Build the TDT table
-----

IF P3$PASS_THREE( START[0],
                  CON_SYM[0, BASE_], CON_CNT,
                  FLD_SYM[0, BASE_],
                  CDX_SYM[0, BASE_], CDX_CNT ) NEQ SUCCESS
THEN
    LEAVE normal;

-----
Pass 4:  parse all /KEY and /DATA specifications.
         define the default record type in temporary
         record definition table.
         Build the KFT table
-----

IF P4$PASS_FOUR( START[0],
                 CON_SYM[0, BASE_], CON_CNT,
                 FLD_SYM[0, BASE_],
                 CDX_SYM[0, BASE_], CDX_CNT ) NEQ SUCCESS
THEN
    LEAVE normal;

!-----
```

```

: 326      0626      | Pass 5:  parse all /include and /omit specifications.
: 327      0627      |          finish the temporary record definition table.
: 328      0628      |          create the record definition table from the
: 329      0629      |          temporary record definition table.
: 330      0630      |          create the constant table.
: 331      0631      |          Build the RDT table
: 332      0632      |-----
: 333      0633      |
: 334      0634      | IF P5$PASS FIVE( START[0],
: 335      0635      |     CON_SYM[0, BASE_], CON_CNT,
: 336      0636      |     FLD_SYM[0, BASE_],
: 337      0637      |     CDX_SYM[0, BASE_], CDX_CNT,
: 338      0638      |     TRDT[0, BASE_] ) NEQ SUCCESS
: 339      0639      | THEN
: 340      0640      |     LEAVE normal;
: 341      0641      |
: 342      0642      | !+++???
: 343      0643      |
: 344      0644      |     It may be worthwhile to determine the following information:
: 345      0645      |     Number of different record formats
: 346      0646      |     Longest (output) record length
: 347      0647      |     Whether any record reformatting is needed
: 348      0648      |     ???
: 349      0649      |
: 350      0650      | !+++???
: 351      0651      |
: 352      0652      | ! Restore the ending address of the work area
: 353      0653      |
: 354      0654      | CA[CA_WRK_END] = .WRK_END;
: 355      0655      |
: 356      0656      | %IF %BLISS( BLISS16 )
: 357      0657      | %THEN
: 358      0658      |
: 359      0659      | ! Check error flag
: 360      0660      |
: 361      0661      | IF .CA[ CA_1ST_SPC_ERR ] NEQ 0
: 362      0662      | THEN ! Error encountered during parse
: 363      0663      |     LEAVE normal;
: 364      0664      |
: 365      0665      | %FI
: 366      0666      |
: 367      0667      | ! No error encountered
: 368      0668      |
: 369      0669      | RETURN SUCCESS;
: 370      0670      |
: 371      0671      | !end normal processing
: 372      0672      | END;
: 373      0673      |
: 374      0674      | ! Invalid specification. Return the first error message
: 375      0675      |
: 376      0676      | %IF %BLISS( BLISS16 )
: 377      0677      | %THEN
: 378      0678      |     !fill the error buffer
: 379      0679      |     BEGIN
: 380      0680      |     LOCAL ERR_BUFF : REF VECTOR[ WORD];
: 381      0681      |     ERR_BUFF = .CA[ CA_STAT_ADR ];
: 382      0682      |     ERR_BUFF[ 0 ] = .CA[ CA_1ST_SPC_ERR ];

```



```
: 383      U 0683 2      ERR_BUFF[ 1 ] = .CA[ CA_1ST_SPC_LIN ];  
: 384      U 0684 2      END;  
: 385      0685 2      %FI  
: 386      0686 2      RETURN FAIL;  
: 387      0687 2  
: 388      0688 2  
: 389      0689 2      !end srtspc  
: 390      0690 1      END;
```

																.TITLE	SRTSPC					
																.IDENT	\V04-000\					
																.PSECT	SORSRO_CODE,NOWRT, SHR, PIC,2					
																P.AAA:	.ASCII	<7>\PROCESS\				
																	.ASCII	<6>\RECORD\				
																	.ASCII	<3>\TAG\				
																	.ASCII	<7>\ADDRESS\				
																	.ASCII	<5>\INDEX\				
																	.ASCII	<6>\STABLE\				
																	.ASCII	<8>\NOSTABLE\				
45	43	4E	45	55	51	45	4C	42	41	54	53	4F	4E	08	00028	.ASCII	<14>\CHECK_SEQUENCE\					
4E	45	55	51	45	53	5F	4B	43	45	48	43	4F	4E	10	00040	.ASCII	<16>\NOCHECK_SEQUENCE\					
																		45	43	0004F		
55	51	45	53	5F	47	4E	49	54	41	4C	4C	4F	43	12	00051	.ASCII	<18>\COLLATING_SEQUENCE\					
																		45	43	00060		
																	.ASCII	<8>\SEQUENCE\				
																	.ASCII	<5>\ASCII\				
																	.ASCII	<6>\EBCDIC\				
4C	41	4E	4F	49	54	41	4E	49	54	4C	55	4D	0D	0007A	.ASCII	<13>\MULTINATIONAL\						
	4E	4F	49	54	41	43	49	46	49	44	4F	4D	0C	00088	.ASCII	<12>\MODIFICATION\						
																	.ASCII	<6>\IGNORE\				
																	.ASCII	<4>\FOLD\				
																	.ASCII	<9>\TIE_BREAK\				
																	.ASCII	<11>\NOTIE_BREAK\				
																	.ASCII	<5>\FIELD\				
																	.ASCII	<4>\NAME\				
																	.ASCII	<4>\SIZE\				
																	.ASCII	<8>\POSITION\				
47	4E	49	54	41	4F	4C	46	5F	49	49	43	53	41	0E	000D7	.ASCII	<14>\ASCII_FLOATING\					
			44	45	4E	4F	5A	5F	49	49	43	53	41	0B	000E6	.ASCII	<11>\ASCII_ZONED\					
																	.ASCII	<6>\BINARY\				
																	.ASCII	<9>\CHARACTER\				
																	.ASCII	<7>\DECIMAL\				
																	.ASCII	<5>\DIBOL\				
																	.ASCII	<10>\F_FLOATING\				
																	.ASCII	<10>\D_FLOATING\				
																	.ASCII	<10>\G_FLOATING\				
																	.ASCII	<10>\H_FLOATING\				
4C	41	4D	49	43	45	44	5F	44	45	4B	43	41	50	0E	0013D	.ASCII	<14>\PACKED_DECIMAL\					
																	.ASCII	<5>\ZONED\				
																	.ASCII	<6>\SIGNED\				
																	.ASCII	<8>\UNSIGNED\				
																	.ASCII	<12>\LEADING_SIGN\				
4E	47	49	53	5F	47	4E	49	4C	49	41	52	54	0D	0016F	.ASCII	<13>\TRAILING_SIGN\						

49	53	5F	44	45	48	43	4E	55	50	52	45	56	4F	10	0017D
	4E	47	49	53	5F	45	54 45	41 44	52 55	41 4C 54	50 43 49 59	45 4E 4D 45 46 48 4C 45 45 43 41 45 44 41 44 45 51 45 45 54 45 54 4E 52 54	4E 53 49 4F 4B 49 54 45 41 44 43 45 44 4F 54 45 4C 4C 47 47 4E 52 4F 44 44 54 44 00	47 0D 07 04 03 02 04 04 09 0A 04 09 04 02 02 02 02 03 02 0A 0D 03 05 00	0018C 0018E 0019C 001A4 001A9 001AD 001B0 001B5 001BA 001C4 001CF 001D4 001DE 001E3 001E6 001E9 001EC 001EF 001F2 001F5 001F9 001FC 00207 00215 00219 0021F
0051	0040	0031	0028	0021	001B	0013	000F	0008	0000						00220
00B7	00AB	00A1	009C	0095	0088	007A	0073	006D	0064						00234
010B	0103	00F9	00F2	00E6	00D7	00D0	00C7	00C2	00BD						00248
016F	0162	0159	0152	014C	013D	0132	0127	011C	0111						0025C
01C4	01BA	01B5	01B0	01AD	01A9	01A4	019C	018E	017D						00270
01F5	01F2	01EF	01EC	01E9	01E6	01E3	01DE	01D4	01CF						00284
				021F	0219	0215	0207	01FC	01F9						00298

```
.ASCII <16>\OVERPUNCHED_SIGN\
.ASCII <13>\SEPARATE_SIGN\
.ASCII <7>\INCLUDE\
.ASCII <4>\OMIT\
.ASCII <3>\KEY\
.ASCII <2>\IF\
.ASCII <4>\THEN\
.ASCII <4>\ELSE\
.ASCII <9>\ASCENDING\
.ASCII <10>\DESCENDING\
.ASCII <4>\DATA\
.ASCII <9>\CONDITION\
.ASCII <4>\TEST\
.ASCII <2>\EQ\
.ASCII <2>\NE\
.ASCII <2>\LE\
.ASCII <2>\LT\
.ASCII <2>\GE\
.ASCII <2>\GT\
.ASCII <3>\AND\
.ASCII <2>\OR\
.ASCII <10>\WORK_FILES\
.ASCII <13>\CDD_PATH_NAME\
.ASCII <3>\PAD\
.ASCII <5>\DTYPE\
.ASCII <0>
```

P.AAB:

WORD 0 8, 15, 19, 27, 33, 40, 49, 64, 81, -
100, 109, 115, 122, 136, 149, 156, 161, -
171, 183, 189, 194, 199, 208, 215, 230, -
242, 249, 259, 267, 273, 284, 295, 306, -
317, 332, 338, 345, 354, 367, 381, 398, -
412, 420, 425, 429, 432, 437, 442, 452, -
463, 468, 478, 483, 486, 489, 492, 495, -
498, 501, 505, 508, 519, 533, 537, 543

SRTWRD=
WRDOFF=

P. AAA
P. AAB

```

.PASS
.EXTRN P1$PASS_ONE, P2$PASS_TWO
.EXTRN P3$PASS_THREE, P4$PASS_FOUR
.EXTRN P5$PASS_FIVE

```

				00FC	00000
	57	0000V	CF	9E	00002
	5E		14	C2	00007
	56	012C	CB	D0	0000A
	50	04	AC	D0	0000F
08	AE		60	7D	00013
10	AE		01	D0	00017
		08	AE	9F	0001B
0000V	CF		01	FB	0001E
			50	D5	00023
			03	12	00025
			00B1	31	00027
	7E	07F8	8F	3C	0002A
	67		01	FB	0002F
	55		50	D0	00032

```

.ENTRY      SOR$$$FPRS, Save R2,R3,R4,R5,R6,R7
MOVAB      SH$SPC HEAP, R7
SUBL2      #20, SP
MOVL       300(CA), WRK_END
MOVL       SPEC_TEXT, R0
MOVQ       (R0), START
MOVL       #1, START+8
PUSHAB     START
CALLS      #1, SI$SKIP_IGNORED
TSTL       R0
BNEQ       1$
BRW        2$
MOVZWL     #2040, -(SP)
CALLS      #1, SH$SPC_HEAP
MOVL       R0, CON_SYM

```

0455
0521
0525
0527
0533

0563

	7E	04	AE	D4	00035	CLRL	CON CNT	: 0564
	67	05FA	8F	3C	00038	MOVZWL	#1530, -(SP)	: 0565
	53		01	FB	0003D	CALLS	#1, SH\$SPC_HEAP	
	7E		50	DO	00040	MOVL	R0, FLD_SYM	
	67	05FA	8F	3C	00043	MOVZWL	#1530, -(SP)	: 0566
	54		01	FB	00048	CALLS	#1, SH\$SPC_HEAP	
			50	DO	0004B	MOVL	R0, CDX_SYM	
	7E		6E	D4	0004E	CLRL	CDX CNT	: 0567
	67	C0	8F	9A	00050	MOVZBL	#192, -(SP)	: 0568
	52		01	FB	00054	CALLS	#1, SH\$SPC_HEAP	
	62		50	DO	00057	MOVL	R0, TRDT	
	62		01	88	0005A	BISB2	#1, (TRDT)	: 0572
	02		02	8A	0005D	BICB2	#2, (TRDT)	: 0573
	A2	00FD	CB	90	00060	MOVB	253(CA), 2(TRDT)	: 0574
		04	AE	9F	00066	PUSHAB	CON CNT	: 0582
			55	DD	00069	PUSHL	CON_SYM	
		10	AE	9F	0006B	PUSHAB	START	
00000000G	00		03	FB	0006E	CALLS	#3, P1\$PASS_ONE	
	01		50	D1	00075	CMPL	R0, #1	
			65	12	00078	BNEQ	3\$	
			53	DD	0007A	PUSHL	FLD_SYM	: 0592
		0C	AE	9F	0007C	PUSHAB	START	
00000000G	00		02	FB	0007F	CALLS	#2, P2\$PASS_TWO	
	01		50	D1	00086	CMPL	R0, #1	
			54	12	00089	BNEQ	3\$	
		4018	8F	BB	0008B	PUSHR	#^M<R3,R4,SP>	: 0607
		10	AE	9F	0008F	PUSHAB	CON CNT	: 0605
			55	DD	00092	PUSHL	CON_SYM	: 0607
		1C	AE	9F	00094	PUSHAB	START	: 0604
00000000G	00		06	FB	00097	CALLS	#6, P3\$PASS_THREE	: 0607
	01		50	D1	0009E	CMPL	R0, #1	
			3C	12	000A1	BNEQ	3\$	
		4018	8F	BB	000A3	PUSHR	#^M<R3,R4,SP>	: 0621
		10	AE	9F	000A7	PUSHAB	CON CNT	: 0619
			55	DD	000AA	PUSHL	CON_SYM	: 0621
		1C	AE	9F	000AC	PUSHAB	START	: 0618
00000000G	00		06	FB	000AF	CALLS	#6, P4\$PASS_FOUR	: 0621
	01		50	D1	000B6	CMPL	R0, #1	
			24	12	000B9	BNEQ	3\$	
			52	DD	000BB	PUSHL	TRDT	: 0638
		04	AE	9F	000BD	PUSHAB	CDX CNT	: 0637
			18	BB	000C0	PUSHR	#^M<R3,R4>	: 0638
		14	AE	9F	000C2	PUSHAB	CON CNT	: 0635
			55	DD	000C5	PUSHL	CON_SYM	: 0638
		20	AE	9F	000C7	PUSHAB	START	: 0634
00000000G	00		07	FB	000CA	CALLS	#7, P5\$PASS_FIVE	: 0638
	01		50	D1	000D1	CMPL	R0, #1	
			09	12	000D4	BNEQ	3\$	
012C	CB		56	DO	000D6	MOVL	WRK_END, 300(CA)	: 0654
	50		01	DO	000DB	MOVL	#1, R0	: 0669
				04	000DE	RET		
			50	D4	000DF	CLRL	R0	: 0687
				04	000E1	RET		: 0690

; Routine Size: 226 bytes, Routine Base: SOR\$R0_CODE + 02A4

SRTSPC
V04-000

: 391

0691 1

B 15
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 12
(3)

SRT
V04


```
393 0692 1 %IF %BLISS(BLISS32) %THEN
394 0693 1 GLOBAL ROUTINE SORS$DEF_FIELD(
395 0694 1     FIELD_NAME:  REF VECTOR[2],
396 0695 1     DATATYPE:   BYTE,
397 0696 1     POSITION:    WORD,
398 0697 1     LENGTH:     WORD,
399 0698 1     SCALE:      BYTE,
400 0699 1     FLD_SYM:    REF SYM_TAB[], ! Field "symbol" table
401 0700 1     KYW_LINE
402 0701 1 ): CA_LINKAGE =
403 0702 2 BEGIN
404 0703 2 LOCAL
405 0704 2     INDEX,
406 0705 2     NAME:  VECTOR[2],
407 0706 2     FDT:  REF FDT_TAB[]; ! Pointer to Field Defn Table
408 0707 2
409 0708 2 ! Get the impure area
410 0709 2
411 0710 2 CA_AREA_( CA );
412 0711 2
413 0712 2 ! Save the name
414 0713 2
415 0714 2 NAME[0] = .(FIELD_NAME[0])<0,16,0>; ! Length is only a word
416 0715 2 NAME[1] = HEAP (.NAME[0]);
417 0716 2 CH$MOVE(.NAME[0], .FIELD_NAME[1], .NAME[1]);
418 0717 2
419 0718 2 FDT = ALLOC (FDT_UNIT);
420 0719 2 CA[CA_FDT_SIZ] = .CA[CA_FDT_SIZ] + 1;
421 0720 2
422 0721 2 ! Too many entries in FDT or field symbol table?
423 0722 2
424 0723 2 IF .CA[CA_FDT_SIZ] GEQ MIN(FDT_MAX, MAX_FIELDS)
425 0724 2 THEN
426 0725 3 BEGIN
427 0726 3     ERR ( .KYW_LINE, SORS$_SPCIVF ); ! Invalid field
428 0727 3     RETURN FAIL;
429 0728 3 END;
430 0729 2
431 0730 2 FDT[0, FDT_TYPE] = .DATATYPE;
432 0731 2 FDT[0, FDT_SCALE] = .SCALE;
433 0732 2 FDT[0, FDT_FLD_POS] = .POSITION;
434 0733 2 FDT[0, FDT_FLD_SIZ] = .LENGTH;
435 0734 2
436 0735 2 ! Look for the name in the field symbol table
437 0736 2
438 0737 2 IF SEARCH TABLE( NAME[0], INDEX,
439 0738 2     FLD_SYM[0,BASE_], .CA[CA_FDT_SIZ]-1 ) EQL SUCCESS
440 0739 2 THEN
441 0740 3 BEGIN ! Name has already been used
442 0741 3     ERR ( .KYW_LINE, SORS$_SPCIVF ); ! Invalid field
443 0742 3     RETURN FAIL;
444 0743 3 END;
445 0744 2
446 0745 2 ! Put the length/address in the field symbol table entry
447 0746 2 ! Put the fdt index in the field symbol table
448 0747 2
449 0748 3 BEGIN
```



```
: 450      0749 3  LOCAL FLD_SYM_LOC: REF SYM_TAB [1];
: 451      0750 3  FLD_SYM_LOC = FLD_SYM[CA[CA_FDT_SIZE]-1, BASE_];
: 452      0751 3  FLD_SYM_LOC[0, SYM_NAM_LEN] = .NAME[LEN];
: 453      0752 3  FLD_SYM_LOC[0, SYM_NAM_ADR] = .NAME[PTR];
: 454      0753 3  FLD_SYM_LOC[0, SYM_INDEX] = .CA[CA_FDT_SIZE]-1;
: 455      0754 2  END;
: 456      0755 2
: 457      0756 2  RETURN SUCCESS;
: 458      0757 2
: 459      0758 1  END;
```

				003C	00000	.ENTRY	SORT\$DEF_FIELD, Save R2,R3,R4,R5	0693
		5E		0C	C2	SUBL2	#12, SP	
		52	04	AC	D0	MOVL	FIELD_NAME, R2	0714
		AE		62	3C	MOVZWL	(R2), NAME	
			04	AE	DD	PUSHL	NAME	0715
	0000V	CF		01	FB	CALLS	#1, SH\$SPC_HEAP	
	08	AE		50	D0	MOVL	R0, NAME+4	
08	BE	04	B2	04	AE	MOVC3	NAME, @4(R2), @NAME+4	0716
				06	DD	PUSHL	#6	0718
	0000V	CF		01	FB	CALLS	#1, SA\$SPC_ALLOC	
		52		50	D0	MOVL	R0, FDT	
			00FF	CB	96	INCB	255(CA)	0719
	FF	8F	00FF	CB	91	CMPB	255(CA), #255	0723
				2D	1E	BGEQU	1\$	
		62	08	AC	90	MOVB	DATATYPE, (FDT)	0730
	01	A2	14	AC	90	MOVB	SCALE, 1(FDT)	0731
	02	A2	0C	AC	B0	MOVW	POSITION, 2(FDT)	0732
	04	A2	10	AC	B0	MOVW	LENGTH, 4(FDT)	0733
		7E	00FF	CB	9A	MOVZBL	255(CA), -(SP)	0738
				6E	D7	DECL	(SP)	
			18	AC	DD	PUSHL	FLD_SYM	
			08	AE	9F	PUSHAB	INDEX	0737
			10	AE	9F	PUSHAB	NAME	
	0000V	CF		04	FB	CALLS	#4, ST\$SEARCH_TABLE	0738
		01		50	D1	CPL	R0, #1	
			001C8194	10	12	BNEQ	2\$	
			1C	8F	DD	PUSHL	#1868180	0741
	0000V	CF		AC	DD	PUSHL	KYW_LINE	
				02	FB	CALLS	#2, EC\$ERR_CNTL	
				21	11	BRB	3\$	0742
		50	00FF	CB	9A	MOVZBL	255(CA), R0	0750
		50		06	C4	MULL2	#6, R0	
		50	18	AC	C0	ADDL2	FLD_SYM, R0	
				50	D7	DECL	FLD_SYM_LOC	
		70	04	AE	90	MOVB	NAME, -(FLD_SYM_LOC)	0751
		70	08	AE	D0	MOVL	NAME+4, -(FLD_SYM_LOC)	0752
05	A0	00FF		01	83	SUBB3	#1, 255(CA), 5(FLD_SYM_LOC)	0753
				01	D0	MOVL	#1, R0	0756
				04	00093	RET		
				50	D4	CLRL	R0	0758
				04	00096	RET		

SRTSPC
V04-000

E 15
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 15
(4)

; Routine Size: 151 bytes, Routine Base: SOR\$RO_CODE + 0386

; 460 0759 1 %FI

```
0760 1 GLOBAL ROUTINE SPC_ALLOC( AMT ): CA_LINKAGE =
0761 1
0762 1 ++
0763 1 Functional Description:
0764 1
0765 1     Allocate some storage from the work area, checking that we don't
0766 1     exceed the end of the work area.
0767 1
0768 1 Formal Parameters:
0769 1
0770 1     AMT      Amount of storage to allocate, in bytes.
0771 1
0772 1 Implicit Inputs:
0773 1
0774 1 Implicit Outputs:
0775 1
0776 1 Routine Value:
0777 1
0778 1     Address of the storage.
0779 1
0780 1 Side Effects:
0781 1
0782 1     None.
0783 1
0784 1 --
0785 2 BEGIN
0786 2 LOCAL
0787 2 T;
0788 2
0789 2 CA_AREA_( CA );
0790 2
0791 2 T = .CA[CA_WRK_ADR];
0792 2 CA[CA_WRK_ADR] = .CA[CA_WRK_ADR] + .AMT;
0793 2 IF .CA[CA_WRK_ADR] GTRA .CA[CA_WRK_END]
0794 2 THEN
0795 3 BEGIN
0796 3 ERR_CNTL(0, SORS_WKAREA, (LOCAL X;X)); ! Insufficient work area
0797 3 RETURN 0;
0798 2 END;
0799 2 RETURN .T;
0800 1 END;
```

```
0004 00000
5E      04 C2 00002
50      CB 9E 00005
52      60 D0 0000A
012C    60 04 AC C0 0000D
        CB 60 D1 00011
        11 1B 00016
        5E DD 00018
        001C815C 8F DD 0001A
        7E D4 00020
0000V CF 03 FB 00022
```

```
.ENTRY SASSPC_ALLOC, Save R2
SUBL2 #4, SP
MOVAB 296(CA), R0
MOVL (R0), T
ADDL2 AMT, (R0)
CMPL (R0), 300(CA)
BLEQU 1$
PUSHL SP
PUSHL #1868124
CLRL -(SP)
CALLS #3, EC$ERR_CNTL
```

```
: 0760
: 0791
: 0792
: 0793
: 0796
:
```


SRTSPC
V04-000

G 15
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 17
(5)

50

04	11	00027		BRB	2\$
52	D0	00029	1\$:	MOVL	T, R0
	04	0002C		RET	
50	D4	0002D	2\$:	CLRL	R0
	04	0002F		RET	

: 0797
: 0799
: 0800
:

; Routine Size: 48 bytes, Routine Base: SOR\$RO_CODE + 041D

```
0801 1 GLOBAL ROUTINE SPC_HEAP( AMT ): CA_LINKAGE =
0802 1
0803 1 !++
0804 1 Functional Description:
0805 1
0806 1     Allocate some temporary storage from the work area, checking that we
0807 1     don't exceed the end of the work area.
0808 1
0809 1 Formal Parameters:
0810 1
0811 1     AMT     Amount of storage to allocate, in bytes.
0812 1
0813 1 Implicit Inputs:
0814 1
0815 1 Implicit Outputs:
0816 1
0817 1 Routine Value:
0818 1
0819 1     Address of the storage.
0820 1
0821 1 Side Effects:
0822 1
0823 1     None.
0824 1
0825 1 --
0826 2 BEGIN
0827 2
0828 2 CA_AREA_( CA );
0829 2
0830 2 CA[CA_WRK_END] = .CA[CA_WRK_END] - .AMT;
0831 2 CA[CA_WRK_END] = .CA[CA_WRK_END] AND NOT 1; ! Word align
0832 2 IF .CA[CA_WRK_ADR] GTRA .CA[CA_WRK_END]
0833 2 THEN
0834 3 BEGIN
0835 3 ERR_CNTL(0, SORS_WKAREA, (LOCAL X;X)); ! Insufficient work area
0836 3 RETURN 0;
0837 2 END;
0838 2 RETURN .CA[CA_WRK_END];
0839 2
0840 1 END;
```

```
0000 00000
5E      04 C2 00002
50      012C CB 9E 00005
60      04 AC C2 0000A
60      01 8A 0000E
60      0128 CB D1 00011
11      1B 00016
5E      DD 00018
001C815C 8F DD 0001A
7E      D4 00020
0000V CF 03 FB 00022
06      11 00027
```

```
.ENTRY SH$SPC_HEAP, Save nothing
SUBL2 #4, SP
MOVAB 300(CA), R0
SUBL2 AMT, (R0)
BICB2 #1, (R0)
CMPL 296(CA), (R0)
BLEQU 1$
PUSHL SP
PUSHL #1868124
CLRL -(SP)
CALLS #3, EC$ERR_CNTL
BRB 2$
```

```
: 0801
:
: 0830
:
: 0831
: 0832
:
: 0835
:
:
: 0836
```


SRTSPC
V04-000

I 15
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 19
(6)

50	012C	CB	D0 00029	1\$:	MOVL	300(CA), R0
			04 0002E		RET	
		50	D4 0002F	2\$:	CLRL	R0
			04 00031		RET	

; 0838
;
;
;
;
;
;

; Routine Size: 50 bytes, Routine Base: SORSRO_CODE + 044D

```
0841 1 GLOBAL ROUTINE CVT_ATB(  
0842 1     BASE,  
0843 1     IN:          REF VECTOR[2],  
0844 1     OUT_ADR:     REF VECTOR[1]) =  
0845 1 ++  
0846 1 Functional Description:  
0847 1     Converts ascii string to positive binary integer  
0848 1  
0849 1 Formal Parameters:  
0850 1  
0851 1     BASE      - Number base (2 - 10 or 16).  
0852 1     IN        - Address of length/address of input string.  
0853 1     OUT_ADR   - Address of word to store resultant binary number.  
0854 1  
0855 1  
0856 1 Implicit Inputs:  
0857 1  
0858 1 Implicit Outputs:  
0859 1  
0860 1 Routine Value:  
0861 1  
0862 1     SUCCESS  
0863 1     FAIL  
0864 1  
0865 1 Side Effects:  
0866 1  
0867 1     None.  
0868 1  
0869 1 --  
0870 2 BEGIN  
0871 2  
0872 2 output_('cvt_atb');  
0873 2  
0874 2 ! Initialize the output value  
0875 2 OUT_ADR[0] = 0;  
0876 2  
0877 2 IF .INC_LEN] LEQ 0 THEN RETURN FAIL;  
0878 2  
0879 2 WHILE TRUE DO  
0880 2     BEGIN  
0881 2         LOCAL  
0882 2             NUM;  
0883 2  
0884 2         ! Get next digit  
0885 2         NUM = LOWER_(CH$RCHAR(.INC_PTR));  
0886 2  
0887 2         ! Convert it to a number  
0888 2         SELECTONE .NUM OF  
0889 2             SET  
0890 2                 [XC'0' TO XC'9']: NUM = .NUM - XC'0';  
0891 2                 [XC'a' TO XC'f']: NUM = .NUM - XC'a' + 10;  
0892 2                 [OTHERWISE]: RETURN FAIL;  
0893 2             TES;  
0894 2  
0895 2         ! Is it in range?  
0896 2         IF .NUM GEQU .BASE THEN RETURN FAIL;  
0897 2
```


602	0898	3
603	0899	3
604	0900	3
605	0901	3
606	0902	3
607	0903	3
608	0904	3
609	0905	3
610	0906	2
611	0907	2
612	0908	2
613	0909	2
614	0910	1

```
! Multiply the result by the radix and add in the number
OUT_ADR[0] = .OUT_ADR[0] * .BASE + .NUM;
```

! Advance to next character

```
IN[_PTR] = .IN[_PTR] + 1;  
IN[_LEN] = .IN[_LEN] - 1;  
IF .IN[_LEN] LEQ 0 THEN EXITLOOP;  
END;
```

RETURN SUCCESS

END:

Address	Hex	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
---------	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

; Routine Size: 87 bytes, Routine Base: SOR\$RO_CODE + 047F

```

616 0911 1 GLOBAL ROUTINE GET_NEXT_SPEC( DESC: REF VECTOR[3], SPC: REF VECTOR[3] ) =
617 0912 1
618 0913 1 !++
619 0914 1 Functional Description:
620 0915 1
621 0916 1     Get the next specification. Specifications have one of the formats:
622 0917 1
623 0918 1     /keyword...=...sub_specification
624 0919 1     /keyword...=...(sub_specification_list)
625 0920 1
626 0921 1     everything beyond the keyword is optional, and each "." represents a
627 0922 1     space, a null, or part of a comment.
628 0923 1     This routine returns the length/address of the sub_specification or
629 0924 1     sub_specification_list.
630 0925 1
631 0926 1 Formal Parameters:
632 0927 1
633 0928 1     DESC      - Address of length/address/line of text to search.
634 0929 1     SPC       - Address to return length/pointer/line of sub_specification
635 0930 1
636 0931 1 Implicit Inputs:
637 0932 1
638 0933 1 Implicit Outputs:
639 0934 1
640 0935 1     None.
641 0936 1
642 0937 1 Routine Value:
643 0938 1
644 0939 1     keyword type
645 0940 1
646 0941 1 Side Effects:
647 0942 1
648 0943 1 --
649 0944 2 BEGIN
650 0945 2
651 0946 2 LOCAL
652 0947 2     KYW_TYPE,
653 0948 2     FIRST_CHAR;
654 0949 2
655 0950 2 output_('get_next_spec');
656 0951 2
657 0952 2 ! Initialize output length
658 0953 2 SPC[LEN] = 0;
659 0954 2
660 0955 2 ! Are we out of input?
661 0956 2 IF .DESC[LEN] EQL 0 THEN RETURN KW_UNKN;
662 0957 2
663 0958 2 ! Save the first character ('/'?)
664 0959 2 FIRST_CHAR = CH$RCHAR(.DESC[PTR]);
665 0960 2
666 0961 2 ! Move off of first character ('/'?)
667 0962 2 DESC[PTR] = .DESC[PTR] + 1;
668 0963 2 DESC[LEN] = .DESC[LEN] - 1;
669 0964 2
670 0965 2 ! Get the sub_specification or sub_spec_list
671 0966 2 ! Note that if first character is not a slash we still want to
672 0967 2 ! call get_sub_spec to move pointers appropriately.
```


SRTSPC
V04-000

M 15
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 23
(8)

```
: 673      0968 2      !
: 674      0969 2      KYW_TYPE = GET_SUB_SPEC( DESC[0], SPC[0] );
: 675      0970 2
: 676      0971 2      IF .FIRST_CHAR NEQ C_SLASH THEN RETURN KW_UNKN;
: 677      0972 2
: 678      0973 2      RETURN .KYW_TYPE
: 679      0974 2
: 680      0975 1      END;
```

			0004 00000	.ENTRY	GNS\$GET_NEXT_SPEC, Save R2	: 0911
		08	BC D4 00002	CLRL	@SPC	: 0953
50		04	AC D0 00005	MOVL	DESC, R0	: 0956
		60	D5 00009	TSTL	(R0)	
		18	13 0000B	BEQL	1\$	
52		04	B0 9A 0000D	MOVZBL	@4(R0), FIRST_CHAR	: 0959
		04	A0 D6 00011	INCL	4(R0)	: 0962
		60	D7 00014	DECL	(R0)	: 0963
		08	AC DD 00016	PUSHL	SPC	: 0969
		50	DD 00019	PUSHL	R0	
0000V	CF	02	FB 0001B	CALLS	#2, GSS\$GET_SUB_SPEC	
	2F	52	D1 00020	CMPL	FIRST_CHAR, #47	: 0971
		04	13 00023	BEQL	2\$	
50		41	8F 9A 00025 1\$:	MOVZBL	#65, R0	
			04 00029 2\$:	RET		: 0975

; Routine Size: 42 bytes, Routine Base: SOR\$R0_CODE + 04D6

```
682 0976 1 GLOBAL ROUTINE GET_SUB_SPEC( SPC: REF VECTOR[3], SUB_SPC: REF VECTOR[3] ) =
683 0977 1
684 0978 1 ++
685 0979 1 Functional Description:
686 0980 1
687 0981 1     Get the next sub_specification. Sub_specifications have the forms:
688 0982 1
689 0983 1     keyword...=...sub_specification
690 0984 1     keyword...=...(sub_specification_list)
691 0985 1
692 0986 1     everything beyond the keyword is optional, and each "." represents a
693 0987 1     space, a null, or part of a comment.
694 0988 1     This routine returns the length/address of the sub_specification or
695 0989 1     sub_specification_list.
696 0990 1
697 0991 1 Formal Parameters:
698 0992 1
699 0993 1     SPC      - Address of length/pointer/line of first character of
700 0994 1               keyword, modified to point to end of sub_specification.
701 0995 1     SUB_SPC  - Address to return length/pointer/line of sub_specification
702 0996 1
703 0997 1 Implicit Inputs:
704 0998 1
705 0999 1 Implicit Outputs:
706 1000 1
707 1001 1     None.
708 1002 1
709 1003 1 Routine Value:
710 1004 1
711 1005 1     keyword type
712 1006 1
713 1007 1 Side Effects:
714 1008 1
715 1009 1 --
716 1010 2 BEGIN
717 1011 2
718 1012 2 LOCAL
719 1013 2     KYW_TYPE;
720 1014 2
721 1015 2     output_('get_sub_spec');
722 1016 2
723 1017 2     ! Initialize output length
724 1018 2     SUB_SPC[LEN] = 0;
725 1019 2
726 1020 2     ! Are we out of input
727 1021 2     IF .SPC[LEN] EQL 0
728 1022 2     THEN
729 1023 2         RETURN KW_UNKN;
730 1024 2
731 1025 2     ! Get the keyword type
732 1026 2     KYW_TYPE = GET_KYW_TYPE( SPC[0] );
733 1027 2
734 1028 2     ! Skip over ignored stuff. If nothing else, just return the keyword.
735 1029 2
736 1030 2     IF SKIP_IGNORED( SPC[0] ) EQL FAIL THEN RETURN .KYW_TYPE;
737 1031 2
738 1032 2     ! Check the character we're sitting on
```



```
: 739      1033 2      IF CH$RCHAR(.SPC[_PTR]) NEQ C_EQUAL
: 740      1034 2      AND
: 741      1035 2      CH$RCHAR(.SPC[_PTR]) NEQ C_COLON
: 742      1036 2      THEN
: 743      1037 2      RETURN .KYW_TYPE;
: 744      1038 2
: 745      1039 2      ! Move pointer off of equal sign
: 746      1040 2      SPC[_PTR] = .SPC[_PTR] + 1;
: 747      1041 2      SPC[_LEN] = .SPC[_LEN] - 1;
: 748      1042 2
: 749      1043 2      ! Skip more ignored stuff
: 750      1044 2      IF SKIP_IGNORED( SPC[0] ) EQL FAIL THEN RETURN KW_UNKN;
: 751      1045 2
: 752      1046 2      ! Are we sitting on a left parenthesis?
: 753      1047 2      IF CH$RCHAR(.SPC[_PTR]) EQL C_L_PAREN
: 754      1048 2      THEN
: 755      1049 2      ! Then its a sub_specification_list
: 756      1050 2      BEGIN
: 757      1051 2
: 758      1052 2      ! Move the pointer off of the left parenthesis
: 759      1053 2      SPC[_PTR] = .SPC[_PTR] + 1;
: 760      1054 2      SPC[_LEN] = .SPC[_LEN] - 1;
: 761      1055 2
: 762      1056 2      ! Skip more ignored stuff
: 763      1057 2      IF SKIP_IGNORED( SPC[0] ) EQL FAIL THEN RETURN KW_UNKN;
: 764      1058 2
: 765      1059 2      ! Start of sub_specification
: 766      1060 2      SUB_SPC[_PTR] = .SPC[_PTR];
: 767      1061 2      SUB_SPC[_LINE] = .SPC[_LINE];
: 768      1062 2
: 769      1063 2      ! Get the corresponding right parenthesis
: 770      1064 2      IF FIND_RPAR( SPC[0] ) EQL FAIL
: 771      1065 2      THEN
: 772      1066 2      RETURN KW_UNKN;
: 773      1067 2
: 774      1068 2      ! Calculate sub_specification_list length
: 775      1069 2      SUB_SPC[_LEN] = .SPC[_PTR] - .SUB_SPC[_PTR];
: 776      1070 2
: 777      1071 2      ! Move off of right parenthesis
: 778      1072 2      SPC[_PTR] = .SPC[_PTR] + 1;
: 779      1073 2      SPC[_LEN] = .SPC[_LEN] - 1;
: 780      1074 2
: 781      1075 2      END
: 782      1076 2
: 783      1077 2      ELSE
: 784      1078 2
: 785      1079 2      BEGIN
: 786      1080 2
: 787      1081 2      SUB_SPC[_LINE] = .SPC[_LINE];
: 788      1082 2
: 789      1083 2      ! Find the end of the sub specification
: 790      1084 2      IF GET_STRING( SPC[0], SUB_SPC[0] ) EQL FAIL
: 791      1085 2      THEN
: 792      1086 2      RETURN KW_UNKN;
: 793      1087 2
: 794      1088 2      END;
: 795      1089 2
```


: 796
: 797
: 798

1090 2
1091 2
1092 1

RETURN .KYW_TYPE
END;

			007C 00000	.ENTRY	GSS\$GET_SUB SPEC, Save R2,R3,R4,R5,R6	: 0976
56	0000V	CF	9E 00002	MOVAB	SI\$SKIP_IGNORED, R6	:
53	04	AC	7D 00007	MOVQ	SPC, R3	: 1021
		64	D4 0000B	CLRL	(R4)	: 1018
		63	D5 0000D	TSTL	(R3)	: 1021
		72	13 0000F	BEQL	3\$	:
		53	DD 00011	PUSHL	R3	: 1026
0000V	CF	01	FB 00013	CALLS	#1, GKT\$GET_KYW_TYPE	:
	55	50	D0 00018	MOVL	R0, KYW_TYPE	:
		53	DD 0001B	PUSHL	R3	: 1030
66		01	FB 0001D	CALLS	#1, SI\$SKIP_IGNORED	:
		50	D5 00020	TSTL	R0	:
		64	13 00022	BEQL	4\$	:
52	04	A3	9E 00024	MOVAB	4(R3), R2	: 1033
3D	00	B2	91 00028	CMPB	@0(R2), #61	:
		06	13 0002C	BEQL	1\$	:
3A	00	B2	91 0002E	CMPB	@0(R2), #58	: 1035
		54	12 00032	BNEQ	4\$	:
		62	D6 00034	INCL	(R2)	: 1040
		63	D7 00036	DECL	(R3)	: 1041
		53	DD 00038	PUSHL	R3	: 1044
66		01	FB 0003A	CALLS	#1, SI\$SKIP_IGNORED	:
		50	D5 0003D	TSTL	R0	:
		42	13 0003F	BEQL	3\$	:
28	00	B2	91 00041	CMPB	@0(R2), #40	: 1047
		2C	12 00045	BNEQ	2\$	:
		62	D6 00047	INCL	(R2)	: 1053
		63	D7 00049	DECL	(R3)	: 1054
		53	DD 0004B	PUSHL	R3	: 1057
66		01	FB 0004D	CALLS	#1, SI\$SKIP_IGNORED	:
		50	D5 00050	TSTL	R0	:
		2F	13 00052	BEQL	3\$	:
04	A4	62	D0 00054	MOVL	(R2), 4(R4)	: 1060
08	A4	A3	D0 00058	MOVL	8(R3), 8(R4)	: 1061
		53	DD 0005D	PUSHL	R3	: 1064
0000V	CF	01	FB 0005F	CALLS	#1, FIND_RPAR	:
		50	D5 00064	TSTL	R0	:
		1B	13 00066	BEQL	3\$	:
64		A4	C3 00068	SUBL3	4(R4), (R2), (R4)	: 1069
		62	D6 0006D	INCL	(R2)	: 1072
		63	D7 0006F	DECL	(R3)	: 1073
		15	11 00071	BRB	4\$	: 1047
08	A4	A3	D0 00073	MOVL	8(R3), 8(R4)	: 1081
		18	BB 00078	PUSHR	#^M<R3,R4>	: 1084
0000V	CF	02	FB 0007A	CALLS	#2, GSS\$GET_STRING	:
		50	D5 0007F	TSTL	R0	:
		05	12 00081	BNEQ	4\$	:
		8F	9A 00083	MOVZBL	#65, R0	: 1086
50	41	04	00087	RET		:

SRTSPC
V04-000

D 16
16-Sep-1984 00:53:54 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:54 [SORT32.SRC]SRTSPCBLI.B32;1

Page 27
(9)

50

55 D0 00088 4\$: MOVL KYW_TYPE, R0
04 0008B RET

: 1090
: 1092

; Routine Size: 140 bytes, Routine Base: SOR\$RO_CODE + 0500


```

: 800      1093 1 ROUTINE FIND_RPAR( SPEC: REF VECTOR[3] ) =
: 801      1094 1
: 802      1095 1 ++
: 803      1096 1 Functional Description:
: 804      1097 1
: 805      1098 1 Find the next right parenthesis that isn't in quotes or comments
: 806      1099 1
: 807      1100 1 Formal Parameters:
: 808      1101 1
: 809      1102 1 SPEC - Address of length/address/line of the buffer.
: 810      1103 1
: 811      1104 1 Implicit Inputs:
: 812      1105 1 None.
: 813      1106 1
: 814      1107 1 Implicit Outputs:
: 815      1108 1
: 816      1109 1 None.
: 817      1110 1
: 818      1111 1 Routine Value:
: 819      1112 1
: 820      1113 1 SUCCESS
: 821      1114 1 FAIL
: 822      1115 1
: 823      1116 1 Side Effects:
: 824      1117 1
: 825      1118 1
: 826      1119 1 --
: 827      1120 2 BEGIN
: 828      1121 2
: 829      1122 2 LOCAL
: 830      1123 2 PAREN_LEVEL, ! Parenthesis nesting level
: 831      1124 2 QUOTE_F, ! Quote flag
: 832      1125 2 COMMENT; ! Comment flag
: 833      1126 2
: 834      1127 2 output_('find_rpar');
: 835      1128 2
: 836      1129 2 PAREN_LEVEL = 0; ! Set the nesting level to zero
: 837      1130 2 QUOTE_F = FALSE; ! Set the quote flag to false
: 838      1131 2 COMMENT = FALSE; ! Set the comment flag to false
: 839      1132 2
: 840      1133 2 ! Find the next right parenthesis
: 841      1134 2 WHILE TRUE DO
: 842      1135 2 BEGIN
: 843      1136 2
: 844      1137 2 ! Are there more characters in input buffer?
: 845      1138 2 IF .SPEC[_LEN] LEQ 0
: 846      1139 2 THEN
: 847      1140 2 RETURN FAIL;
: 848      1141 2
: 849      1142 2 ! Check the character we're sitting on
: 850      1143 2 SELECT ONE CH$RCHAR(.SPEC[_PTR]) OF
: 851      1144 2 SET
: 852      1145 2
: 853      1146 2 [C_QUOTE]: ! A quote
: 854      1147 2
: 855      1148 2 BEGIN
: 856      1149 2
```



```

: 857      1150  4      ! Are we in a comment
: 858      1151  4      IF NOT .COMMENT
: 859      1152  4      THEN
: 860      1153  4          ! Toggle the quote flag
: 861      1154  4          QUOTE_F = NOT .QUOTE_F;
: 862      1155  4
: 863      1156  3      END;
: 864      1157  3
: 865      1158  3      [C_NULL, C_LF]:      ! Null or line feed indicates end of line
: 866      1159  4      BEGIN
: 867      1160  4      IF NOT .QUOTE_F
: 868      1161  4      THEN
: 869      1162  5          BEGIN
: 870      1163  5              SPEC[LINE] = .SPEC[LINE] + 1;      ! Bump line number
: 871      1164  5              COMMENT = FALSE;      ! Not in a comment
: 872      1165  4          END;
: 873      1166  3      END;
: 874      1167  3
: 875      1168  3      [C_EXCLAM]:      ! Exclamation point
: 876      1169  4      BEGIN
: 877      1170  4      IF NOT .QUOTE_F
: 878      1171  4      THEN
: 879      1172  4          COMMENT = TRUE;      ! Set the comment flag to true
: 880      1173  3      END;
: 881      1174  3
: 882      1175  3      [C_L_PAREN]:      ! A left parenthesis
: 883      1176  3      ! If the comment flag and quote flag are both off then
: 884      1177  3      ! Increment the nesting level
: 885      1178  3      IF NOT .COMMENT AND NOT .QUOTE_F
: 886      1179  3      THEN
: 887      1180  3          PAREN_LEVEL = .PAREN_LEVEL + 1;
: 888      1181  3
: 889      1182  3      [C_R_PAREN]:      ! A right parenthesis
: 890      1183  4      BEGIN
: 891      1184  4      ! If both the comment flag and quote flag are off and the
: 892      1185  4      ! Nesting level is zero then we're done
: 893      1186  4      IF NOT .COMMENT AND NOT .QUOTE_F
: 894      1187  4      THEN
: 895      1188  4          ! Reduce the nesting level, checking for the end
: 896      1189  4          IF (PAREN_LEVEL = .PAREN_LEVEL - 1) LSS 0
: 897      1190  4          THEN
: 898      1191  4              RETURN SUCCESS;
: 899      1192  3      END;
: 900      1193  3
: 901      1194  3      TES;
: 902      1195  3
: 903      1196  3      ! Increment pointer and decrement length
: 904      1197  3      SPEC[PTR] = .SPEC[PTR] + 1;
: 905      1198  3      SPEC[LEN] = .SPEC[LEN] - 1;
: 906      1199  3
: 907      1200  2      END;
: 908      1201  2
: 909      1202  2      RETURN FAIL
: 910      1203  2
: 911      1204  1      END;
```



```
001C 00000 FIND_RPAR:
      .WORD      Save R2,R3,R4
52    D4 00002   CLRL      QUOTE_F      : 1093
53    7C 00004   CLRL      COMMENT      : 1130
50    04 AC D0 00006   MOVL      SPEC, R0      : 1131
60    D5 0000A 1$:   TSTL      (R0)          : 1138
59    15 0000C   BLEQ      8$
51    04 B0 9A 0000E   MOVZBL   @4(R0), R1      : 1143
22    51 91 00012   CMPB      R1, #34         : 1146
08    12 00015   BNEQ      2$
46    53 E8 00017   BLBS      COMMENT, 7$      : 1151
52    52 D2 0001A   MCOML     QUOTE_F, QUOTE_F : 1154
41    11 0001D   BRB        7$              : 1143
51    D5 0001F 2$:   TSTL      R1             : 1158
05    13 00021   BEQL      3$
0A    51 91 00023   CMPB      R1, #10         :
0A    12 00026   BNEQ      4$
35    52 E8 00028 3$: BLBS      QUOTE_F, 7$      : 1160
08    A0 D6 0002B   INCL      8(R0)          : 1163
53    D4 0002E   CLRL      COMMENT          : 1164
2E    11 00030   BRB        7$              : 1143
21    51 91 00032 4$: CMPB      R1, #33         : 1168
08    12 00035   BNEQ      5$
26    52 E8 00037   BLBS      QUOTE_F, 7$      : 1170
53    01 D0 0003A   MOVL      #1, COMMENT      : 1172
21    11 0003D   BRB        7$              : 1143
28    51 91 0003F 5$: CMPB      R1, #40         : 1175
0A    12 00042   BNEQ      6$
19    53 E8 00044   BLBS      COMMENT, 7$      : 1178
16    52 E8 00047   BLBS      QUOTE_F, 7$      :
54    D6 0004A   INCL      PAREN_LEVEL      : 1180
12    11 0004C   BRB        7$              : 1178
29    51 91 0004E 6$: CMPB      R1, #41         : 1182
0D    12 00051   BNEQ      7$
0A    53 E8 00053   BLBS      COMMENT, 7$      : 1186
07    52 E8 00056   BLBS      QUOTE_F, 7$      :
04    54 F4 00059   SOBGEQ    PAREN_LEVEL, 7$ : 1189
50    01 D0 0005C   MOVL      #1, R0          : 1191
      04 0005F   RET
04    A0 D6 00060 7$: INCL      4(R0)          : 1197
60    D7 00063   DECL      (R0)            : 1198
A3    11 00065   BRB        1$              : 1134
50    D4 00067 8$:   CLRL      R0            : 1204
      04 00069   RET
```

; Routine Size: 106 bytes, Routine Base: SOR\$RO_CODE + 058C


```
913 1205 1 GLOBAL ROUTINE SKIP_IGNORED( DESC: REF VECTOR[3] ) =
914 1206 1
915 1207 1 ++
916 1208 1 Functional Description:
917 1209 1
918 1210 1     Get the next character, ignoring blanks, nulls, and comments
919 1211 1
920 1212 1 Formal Parameters:
921 1213 1
922 1214 1     DESC    - Address of length/address/line to search.
923 1215 1
924 1216 1 Implicit Inputs:
925 1217 1
926 1218 1 Implicit Outputs:
927 1219 1
928 1220 1 Routine Value:
929 1221 1
930 1222 1     SUCCESS
931 1223 1     FAIL    - If run out of buffer before significant character is found.
932 1224 1
933 1225 1 Side Effects:
934 1226 1
935 1227 1 --
936 1228 2 BEGIN
937 1229 2
938 1230 2 LOCAL
939 1231 2 COMMENT;                ! Comment flag
940 1232 2
941 1233 2 output_('skip_ignored');
942 1234 2
943 1235 2 COMMENT = FALSE;
944 1236 2
945 1237 2 ! Skip around all the ignored stuff
946 1238 2 WHILE TRUE DO
947 1239 3 BEGIN
948 1240 3
949 1241 3 ! Are there more characters in input buffer?
950 1242 3 IF .DESC[LEN] LEQ 0
951 1243 3 THEN
952 1244 3 RETURN FAIL;
953 1245 3
954 1246 3 ! Check the character we're sitting on
955 1247 3 SELECT ONE CH$RCHAR(.DESC[PTR]) OF
956 1248 3 SET
957 1249 3
958 1250 3 [C_SPACE, C_TAB, C_CR]:    ! Space, tab, or carriage return
959 1251 3 0;                        ! Do nothing
960 1252 3
961 1253 3 [C_NULL, C_LF]:           ! Null, line feed indicates end of line
962 1254 4 BEGIN
963 1255 4 DESC[LINE] = .DESC[LINE] + 1;    ! Bump the line number
964 1256 4 COMMENT = FALSE;                ! Not in a comment
965 1257 3 END;
966 1258 3
967 1259 3 [C_EXCLAM]:                ! Exclamation point
968 1260 3 COMMENT = TRUE;           ! In a comment
969 1261 3
```



```
: 970      1262      3
: 971      1263      3
: 972      1264      3
: 973      1265      3
: 974      1266      3
: 975      1267      3
: 976      1268      3
: 977      1269      3
: 978      1270      3
: 979      1271      3
: 980      1272      3
: 981      1273      3
: 982      1274      3
: 983      1275      1
```

```
[OTHERWISE]:      ! Any other character
      IF NOT .COMMENT THEN RETURN SUCCESS;

      TES;

      ! Increment pointer and decrement N
      DESC[_PTR] = .DESC[_PTR] + 1;
      DESC[_LEN] = .DESC[_LEN] - 1;

      END;

      RETURN FAIL

      END;
```

		0004 00000	.ENTRY	SI\$SKIP_IGNORED, Save R2	
		52 D4 00002	CLRL	COMMENT	1205
50	04	AC D0 00004	MOVL	DESC, R0	1235
		60 D5 00008 1\$:	TSTL	(R0)	1242
		3B 15 0000A	BLEQ	6\$	
51	04	B0 9A 0000C	MOVZBL	@4(R0), R1	1247
09		51 91 00010	CMPB	R1, #9	1250
		2B 13 00013	BEQL	5\$	
0D		51 91 00015	CMPB	R1, #13	
		26 13 00018	BEQL	5\$	
20		51 91 0001A	CMPB	R1, #32	
		21 13 0001D	BEQL	5\$	
		51 D5 0001F	TSTL	R1	1253
		05 13 00021	BEQL	2\$	
0A		51 91 00023	CMPB	R1, #10	
		07 12 00026	BNEQ	3\$	
	08	A0 D6 00028 2\$:	INCL	8(R0)	1255
		52 D4 0002B	CLRL	COMMENT	1256
		11 11 0002D	BRB	5\$	1247
21		51 91 0002F 3\$:	CMPB	R1, #33	1259
		05 12 00032	BNEQ	4\$	
52		01 D0 00034	MOVL	#1, COMMENT	1260
		07 11 00037	BRB	5\$	
04		52 E8 00039 4\$:	BLBS	COMMENT, 5\$	1263
50		01 D0 0003C	MOVL	#1, R0	
		04 0003F	RET		
	04	A0 D6 00040 5\$:	INCL	4(R0)	1268
		60 D7 00043	DECL	(R0)	1269
		C1 11 00045	BRB	1\$	1238
		50 D4 00047 6\$:	CLRL	R0	1275
		04 00049	RET		

; Routine Size: 74 bytes, Routine Base: SOR\$RO_CODE + 05F6


```

: 985      1276 1 GLOBAL ROUTINE SKIP_COMMA( DESC: REF VECTOR[3] ) =
: 986      1277 1
: 987      1278 1 ++
: 988      1279 1 Functional Description:
: 989      1280 1
: 990      1281 1     Get the next character, ignoring blanks, nulls, comments, and commas
: 991      1282 1
: 992      1283 1 Formal Parameters:
: 993      1284 1
: 994      1285 1     DESC    - Address of length/address/line to search.
: 995      1286 1
: 996      1287 1 Implicit Inputs:
: 997      1288 1     None.
: 998      1289 1
: 999      1290 1 Implicit Outputs:
1000      1291 1
1001      1292 1     None.
1002      1293 1
1003      1294 1 Routine Value:
1004      1295 1
1005      1296 1     SUCCESS
1006      1297 1     FAIL
1007      1298 1
1008      1299 1 Side Effects:
1009      1300 1
1010      1301 1 --
1011      1302 2 BEGIN
1012      1303 2
1013      1304 2 LOCAL
1014      1305 2     FLAG;           ! Found, not found flag
1015      1306 2
1016      1307 2 output_('skip_comma');
1017      1308 2
1018      1309 2 IF SKIP_IGNORED( DESC[0] ) EQL FAIL THEN RETURN FAIL;
1019      1310 2
1020      1311 2 IF CH$RCHAR(.DESC[_PTR]) NEQ C_COMMA THEN RETURN FAIL;
1021      1312 2
1022      1313 2 DESC[_PTR] = .DESC[_PTR] + 1;
1023      1314 2 DESC[_LEN] = .DESC[_LEN] - 1;
1024      1315 2
1025      1316 2 IF SKIP_IGNORED( DESC[0] ) EQL FAIL THEN RETURN FAIL;
1026      1317 2
1027      1318 2 RETURN SUCCESS;
1028      1319 2
1029      1320 1 END;
```

				0004 00000	.ENTRY	SC\$SKIP_COMMA, Save R2	: 1276
	52	04	AC	D0 00002	MOVL	DESC, R2	: 1309
			52	DD 00006	PUSHL	R2	:
AA	AF		01	FB 00008	CALLS	#1, SI\$SKIP_IGNORED	:
			50	D5 0000C	TSTL	R0	:
			19	13 0000E	BEQL	1\$	:
	2C	04	B2	91 00010	CMPB	@4(R2), #44	: 1311

SRTSPC
V04-000

K 16
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 34
(12)

		04	13	12	00014	BNEQ	1\$	:	
			A2	D6	00016	INCL	4(R2)	:	1313
			62	D7	00019	DECL	(R2)	:	1314
			52	DD	0001B	PUSHL	R2	:	1316
95	AF		01	FB	0001D	CALLS	#1, SI\$SKIP_IGNORED	:	
			50	D5	00021	TSTL	R0	:	
			04	13	00023	BEQL	1\$	:	
	50		01	D0	00025	MOVL	#1, R0	:	1318
				04	00028	RET		:	
			50	D4	00029	CLRL	R0	:	1320
				04	0002B	RET		:	
								:	

; Routine Size: 44 bytes, Routine Base: SOR\$RO_CODE + 0640


```
1031 1321 1 GLOBAL ROUTINE GET_KYW_TYPE( DESC: REF VECTOR[3] ) =
1032 1322 1
1033 1323 1 ++
1034 1324 1 Functional Description:
1035 1325 1
1036 1326 1     Get the next keyword
1037 1327 1
1038 1328 1 Formal Parameters:
1039 1329 1
1040 1330 1     DESC - Address of length/address
1041 1331 1
1042 1332 1 Implicit Inputs:
1043 1333 1
1044 1334 1     None.
1045 1335 1
1046 1336 1 Implicit Outputs:
1047 1337 1
1048 1338 1     None.
1049 1339 1
1050 1340 1 Routine Value:
1051 1341 1
1052 1342 1     Code for specification file keyword or KW_UNKN
1053 1343 1
1054 1344 1 Side Effects:
1055 1345 1
1056 1346 1 --
1057 1347 2 BEGIN
1058 1348 2
1059 1349 2 LOCAL
1060 1350 2     STRING: VECTOR[3],      ! Local constant length/address to input string
1061 1351 2     KYW_TYPE;              ! Keyword type
1062 1352 2
1063 1353 2 output_('get_kyw_type');
1064 1354 2
1065 1355 2 ! Get the keyword string
1066 1356 2 IF GET_STRING( DESC[0], STRING[0] ) EQL FAIL
1067 1357 2 THEN
1068 1358 2     RETURN KW_UNKN;
1069 1359 2
1070 1360 2 KYW_TYPE = KW_UNKN;
1071 1361 2
1072 1362 2 ! Run through all the spec file keywords
1073 1363 2 !
1074 1364 2 INCR IN FROM 0 TO KW_UNKN-1 DO
1075 1365 2     BEGIN
1076 1366 2     LOCAL
1077 1367 2         KYW: REF VECTOR[BYTE],      ! Address of a keyword
1078 1368 2         PTR: REF VECTOR[BYTE];      ! Local pointer to input string
1079 1369 2
1080 1370 2     ! Initialize the local pointer to input string
1081 1371 2     PTR = .STRING[_PTR];
1082 1372 2
1083 1373 2     ! Get the address of the next keyword
1084 1374 2     KYW = SRTWRD[.WRDOFF[.IN]];
1085 1375 2
1086 1376 2     ! Is the input string shorter than the keyword
1087 1377 2     IF .STRING[_LEN] LEQU CH$RCHAR_A( KYW )
```



```

: 1088      1378      3      THEN
: 1089      1379      4      BEGIN
: 1090      1380      4
: 1091      1381      4      ! Do the keyword and the input string,
: 1092      1382      4      ! (converted to upper case) match?
: 1093      1383      4
: 1094      1384      5      IF BEGIN
: 1095      1385      5      DEC COUNT FROM .STRING[LEN]-1 TO 0 DO
: 1096      1386      6      IF LOWER(.PTR[.COUNT]) NEQ LOWER_(.KYW[.COUNT])
: 1097      1387      5      THEN EXITLOOP 0
: 1098      1388      5      END
: 1099      1389      4      THEN
: 1100      1390      5      BEGIN
: 1101      1391      5
: 1102      1392      5      ! We are done if it was an exact match
: 1103      1393      5      ! Note that we must check for an exact match before checking
: 1104      1394      5      ! for ambiguous keywords, since some keywords (namely, ASCII
: 1105      1395      5      ! and LE) may be prefixes for other keywords.
: 1106      1396      5      IF .STRING[LEN] EQL .KYW[LEN] THEN RETURN .IN;
: 1107      1397      5
: 1108      1398      5      ! Ambiguous keyword?
: 1109      1399      5      IF .KYW_TYPE NEQ KW_UNKN THEN RETURN KW_UNKN;
: 1110      1400      5
: 1111      1401      5      ! Set the keyword type
: 1112      1402      5      KYW_TYPE = .IN;
: 1113      1403      5
: 1114      1404      4      END;
: 1115      1405      4      END;
: 1116      1406      3      END;
: 1117      1407      3
: 1118      1408      2      END;
: 1119      1409      2
: 1120      1410      2      ! Return the keyword
: 1121      1411      2
: 1122      1412      2      RETURN .KYW_TYPE;
: 1123      1413      2
: 1124      1414      1      END;
```

```

                                007C 00000
                                0C C2 00002
                                5E DD 00005
                                04 AC DD 00007
0000V CF                        02 FB 0000A
                                50 D5 0000F
                                50 13 00011
                                56 41 8F 9A 00013
                                54 D4 00017
                                53 04 AE D0 00019 1$:
                                50 F973 CF 9E 0001D
                                51 FB8D CF44 3C 00022
                                51 50 C0 00028
                                50 81 9A 0002B
                                50 6E D1 0002E

.ENTRY GKT$GET_KYW_TYPE, Save R2,R3,R4,R5,R6
SUBL2 #12, SP
PUSHL SP
PUSHL DESC
CALLS #2, GS$GET_STRING
TSTL R0
BEQL 5$
MOVZBL #65, KYW_TYPE
CLRL IN
MOVL STRING+4, PTR
MOVAB SRTWRD, R0
MOVZWL WRDOFF[IN], KYW
ADDL2 R0, KYW
MOVZBL (KYW)+, R0
CMPL STRING, R0
```

```

: 1321
: 1356
:
: 1360
: 1377
: 1371
: 1374
:
: 1377
:
```


			38	1A	00031	BGTRU	7\$		
		50	6E	D0	00033	MOVL	STRING, COUNT		1386
			13	11	00036	BRB	3\$		
		55	6043	9A	00038	2\$:	MOVZBL	(COUNT)[PTR], R5	
		55	20	C8	0003C		BISL2	#32, R5	
		52	6041	9A	0003F		MOVZBL	(COUNT)[KYW], R2	
		52	20	C8	00043		BISL2	#32, R2	
		52	55	D1	00046		CMPL	R5, R2	
			20	12	00049		BNEQ	7\$	
		EA	50	F4	0004B	3\$:	SOBGEQ	COUNT, 2\$	
6E	FF	A1	08	00	ED	0004E	CMPL	#0, #8, -1(KYW), STRING	1396
			04	12	00054		BNEQ	4\$	
		50	54	D0	00056		MOVL	IN, R0	
			04	04	00059		RET		
		00000041	8F	56	D1	0005A	4\$:	CMPL	KYW_TYPE, #65
			05	13	00061		BEQL	6\$	1399
		50	41	8F	9A	00063	5\$:	MOVZBL	#65, R0
				04	00067		RET		
		56	54	D0	00068	6\$:	MOVL	IN, KYW_TYPE	1402
	A6	54	00000040	8F	F3	0006B	7\$:	AOBLEQ	#64, IN, 1\$
		50	56	D0	00073		MOVL	KYW_TYPE, R0	1412
			04	04	00076		RET		1414

; Routine Size: 119 bytes, Routine Base: SOR\$RO_CODE + 066C


```
: 1126      1415 1 GLOBAL ROUTINE GET_STRING( DESC: REF VECTOR[3], STRING: REF VECTOR[3] ) =
: 1127      1416 1
: 1128      1417 1 ++
: 1129      1418 1 Functional Description:
: 1130      1419 1
: 1131      1420 1     Get the string of contiguous significant characters in DESC.
: 1132      1421 1     (Anything inside quotes is significant.)
: 1133      1422 1
: 1134      1423 1 Formal Parameters:
: 1135      1424 1
: 1136      1425 1     DESC    - Address of length/address to search
: 1137      1426 1     STRING  - Address of stored length/address of string.
: 1138      1427 1
: 1139      1428 1 Implicit Inputs:
: 1140      1429 1
: 1141      1430 1     None.
: 1142      1431 1
: 1143      1432 1 Implicit Outputs:
: 1144      1433 1
: 1145      1434 1     None.
: 1146      1435 1
: 1147      1436 1 Routine Value:
: 1148      1437 1
: 1149      1438 1     SUCCESS
: 1150      1439 1     FAIL
: 1151      1440 1
: 1152      1441 1 Side Effects:
: 1153      1442 1
: 1154      1443 1 --
: 1155      1444 2 BEGIN
: 1156      1445 2 LOCAL
: 1157      1446 2     QUOTED,           ! Are we in a quote
: 1158      1447 2     LENGTH;          ! Length of the found string
: 1159      1448 2
: 1160      1449 2 output_('get_string');
: 1161      1450 2
: 1162      1451 2 QUOTED = FALSE;
: 1163      1452 2
: 1164      1453 2 ! Initialize length
: 1165      1454 2 LENGTH = 0;
: 1166      1455 2
: 1167      1456 2 ! Set pointer to string
: 1168      1457 2 STRING[_PTR] = .DESC[_PTR];
: 1169      1458 2 STRING[_LINE] = .DESC[_LINE];
: 1170      1459 2
: 1171      1460 2 ! Find the string
: 1172      1461 2 WHILE TRUE DO
: 1173      1462 2 BEGIN
: 1174      1463 2
: 1175      1464 2     IF .DESC[_LEN] LEQ 0 THEN EXITLOOP;           ! Out of input
: 1176      1465 2
: 1177      1466 2     ! Look at the next character
: 1178      1467 2     SELECTONE CH$RCHAR(.DESC[_PTR]) OF
: 1179      1468 2     SET
: 1180      1469 2     [C_QUOTE]:
: 1181      1470 2     QUOTED = NOT .QUOTED;
: 1182      1471 2     [X'41' TO X'5A', X'61' TO X'7A',
```



```
: 1183 1472 3
: 1184 1473 3
: 1185 1474 3
: 1186 1475 3
: 1187 1476 3
: 1188 1477 3
: 1189 1478 3
: 1190 1479 3
: 1191 1480 3
: 1192 1481 3
: 1193 1482 3
: 1194 1483 3
: 1195 1484 2
: 1196 1485 2
: 1197 1486 2
: 1198 1487 2
: 1199 1488 2
: 1200 1489 2
: 1201 1490 2
: 1202 1491 2
: 1203 1492 2
: 1204 1493 2
: 1205 1494 1
```

```
      %X'30' TO %X'39', %C'_' , %C'$' , C_PERCENT]:
      0;
      [OTHERWISE]:
      IF NOT .QUOTED THEN EXITLOOP;      ! Inside of a quote?
      TES;

      ! Update the values
      DESC[_PTR] = .DESC[_PTR] + 1;
      DESC[_LEN] = .DESC[_LEN] - 1;

      LENGTH = .LENGTH + 1;

      END;

      IF .QUOTED THEN RETURN FAIL;

      IF (STRING[_LEN] = .LENGTH) EQL 0
      THEN
      RETURN FAIL
      ELSE
      RETURN SUCCESS;

      END;
```

			001C 00000	.ENTRY GSSGET_STRING, Save R2,R3,R4	: 1415
			53 7C 00002	CLRQ QUOTED	: 1451
04	51	04	AC 7D 00004	MOVQ DESC, R1	: 1457
	A2	04	A1 7D 00008	MOVQ 4(R1), 4(R2)	
			61 D5 0000D 1\$:	TSTL (R1)	: 1464
			4C 15 0000F	BLEQ 8\$	
	50	04	B1 9A 00011	MOVZBL 24(R1), R0	: 1467
	22		50 91 00015	CMPB R0, #34	: 1469
			05 12 00018	BNEQ 2\$	
	53		53 D2 0001A	MCOML QUOTED, QUOTED	: 1470
			35 11 0001D	BRB 7\$	
	24		50 91 0001F 2\$:	CMPB R0, #36	: 1471
			05 1F 00022	BLSSU 3\$	
	25		50 91 00024	CMPB R0, #37	
			2B 1B 00027	BLEQU 7\$	
	30		50 91 00029 3\$:	CMPB R0, #48	
			05 1F 0002C	BLSSU 4\$	
	39		50 91 0002E	CMPB R0, #57	
			21 1B 00031	BLEQU 7\$	
41	8F		50 91 00033 4\$:	CMPB R0, #65	
			06 1F 00037	BLSSU 5\$	
5A	8F		50 91 00039	CMPB R0, #90	
			15 1B 0003D	BLEQU 7\$	
5F	8F		50 91 0003F 5\$:	CMPB R0, #95	
			0F 13 00043	BEQL 7\$	
61	8F		50 91 00045	CMPB R0, #97	
			06 1F 00049	BLSSU 6\$	
7A	8F		50 91 0004B	CMPB R0, #122	
			03 1B 0004F	BLEQU 7\$	

SRTSPC
V04-000

E 1
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 40
(14)

0C		53	E9	00051	6\$:	BLBC	QUOTED, 9\$	:	1475
	04	A1	D6	00054	7\$:	INCL	4(R1)	:	1479
		61	D7	00057		DECL	(R1)	:	1480
		54	D6	00059		INCL	LENGTH	:	1482
		80	11	0005B		BRB	1\$	:	1461
09		53	E8	0005D	8\$:	BLBS	QUOTED, 10\$	:	1486
62		54	D0	00060	9\$:	MOVL	LENGTH, (R2)	:	1488
		04	13	00063		BEQL	10\$	:	
50		01	D0	00065		MOVL	#1, R0	:	1492
			04	00068		RET		:	
		50	D4	00069	10\$:	CLRL	R0	:	1494
			04	0006B		RET		:	

; Routine Size: 108 bytes, Routine Base: SOR\$RO_CODE + 06E3


```
1207 1495 1 GLOBAL ROUTINE GET_CHAR_CLAUSE(  
1208 1496 1     DESC: REF VECTOR[3],  
1209 1497 1     CH4: REF VECTOR[4, BYTE] ) =  
1210 1498 1  
1211 1499 1 ++  
1212 1500 1     Functional Description:  
1213 1501 1  
1214 1502 1         Get the character clause within DESC.  
1215 1503 1  
1216 1504 1         This is of the form:  
1217 1505 1         { s | d | s dash s | { s | d } eq { s | d } | { s | d } { lt | gt } s }  
1218 1506 1  
1219 1507 1     Formal Parameters:  
1220 1508 1  
1221 1509 1         DESC    Address of length/address of the text.  
1222 1510 1         CH4     Address to place up to four characters (0 thru 3).  
1223 1511 1         0-1 = First character.  
1224 1512 1             Single character or  
1225 1513 1             first character of double character or  
1226 1514 1             first character of range.  
1227 1515 1         2-3 = Second character.  
1228 1516 1             Same as first character if return value is SINGLE, or  
1229 1517 1             second character of definition or  
1230 1518 1             second character of range.  
1231 1519 1  
1232 1520 1     Implicit Inputs:  
1233 1521 1  
1234 1522 1         None.  
1235 1523 1  
1236 1524 1     Implicit Outputs:  
1237 1525 1  
1238 1526 1         None.  
1239 1527 1  
1240 1528 1     Routine Value:  
1241 1529 1  
1242 1530 1         FAIL          invalid clause  
1243 1531 1         GC_SINGLE     <single>  
1244 1532 1         GC_DOUBLE    <double>  
1245 1533 1         GC_S_TO_S     <single> - <single>  
1246 1534 1         GC_x_op_y  
1247 1535 1  
1248 1536 1     Side Effects:  
1249 1537 1  
1250 1538 1 --  
1251 1539 2     BEGIN  
1252 1540 2  
1253 1541 2     LOCAL  
1254 1542 2         CHAR1, CHAR2, OP;  
1255 1543 2  
1256 1544 2     %IF GC_SINGLE NEQ 1 %THEN %WARN('Error in GET_CHAR_CLAUSE') %FI  
1257 1545 2     %IF GC_DOUBLE NEQ 2 %THEN %WARN('Error in GET_CHAR_CLAUSE') %FI  
1258 1546 2  
1259 1547 2     MACRO  
1260 1548 2         ADVANCE (X) =  
1261 1549 2         BEGIN  
1262 1550 2             X[_PTR] = .X[_PTR] + 1;  
1263 1551 2             X[_LEN] = .X[_LEN] - 1;
```



```

END %;

output_('get_char_clause');

! Get the first character(s)
CHAR1 = GET ONE CHAR( DESC[0], CH4[0] );
CH4[2] = .CH4[0];
CH4[3] = .CH4[1];

IF .CHAR1 EQL FAIL THEN RETURN FAIL;

IF SKIP_IGNORED( DESC[0] ) EQL FAIL THEN RETURN .CHAR1;

SELECT CH$RCHAR(.DESC[_PTR]) OF
SET
  [C_DASH, C_EQUAL, C_LESS, C_GREATER]:    ! Need another character(s)
  BEGIN
    ADVANCE(DESC);      ! Move off of character
    CHAR2 = GET ONE CHAR( DESC[0], CH4[2] );
    IF .CHAR2 EQL FAIL THEN RETURN FAIL;
  END;
[OTHERWISE]:
  RETURN .CHAR1;
[C_DASH]:
  BEGIN
    IF NOT (.CHAR1 EQL 1 AND .CHAR2 EQL 1) THEN RETURN FAIL;
    RETURN GC_S_TO_S;
  END;
[C_LESS]:
  OP = %B'11';
[C_EQUAL]:
  OP = %B'00';
[C_GREATER]:
  OP = %B'01';
[C_LESS, C_EQUAL, C_GREATER]:
  RETURN GC_(.CHAR1,.OP,.CHAR2);
TES;

RETURN FAIL;

END;
```

PC	Op	OpC	OpD	OpF	OpI	OpJ	OpK	OpL	OpM	OpN	OpO	OpP	OpQ	OpR	OpS	OpT	OpU	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

	54	04	2F 13 00021	BEQL	3\$		
	55		B2 9A 00023	MOVZBL	24(R2), R4		1565
	2D		01 D0 00027	MOVL	#1, R5		
			54 91 0002A	CMPB	R4, #45		1567
			0A 13 0002D	BEQL	1\$		
	3C		54 91 0002F	CMPB	R4, #60		
			1B 1F 00032	BLSSU	2\$		
	3E		54 91 00034	CMPB	R4, #62		
			16 1A 00037	BGTRU	2\$		
			55 D4 00039 1\$:	CLRL	R5		
		04	A2 D6 0003B	INCL	4(R2)		1569
			62 D7 0003E	DECL	(R2)		
		02	A3 9F 00040	PUSHAB	2(R3)		1570
			52 DD 00043	PUSHL	R2		
0000V	CF		02 FB 00045	CALLS	#2, GOC\$GET_ONE_CHAR		
	53		50 D0 0004A	MOVL	R0, CHAR2		
			47 13 0004D	BEQL	9\$		1571
	04		55 E9 0004F 2\$:	BLBC	R5, 4\$		1573
	50		56 D0 00052 3\$:	MOVL	CHAR1, R0		1574
			04 00055	RET			
	2D		54 91 00056 4\$:	CMPB	R4, #45		1575
			0E 12 00059	BNEQ	5\$		
	01		56 D1 0005B	CMPL	CHAR1, #1		1577
			36 12 0005E	BNEQ	9\$		
	01		53 D1 00060	CMPL	CHAR2, #1		
			31 12 00063	BNEQ	9\$		
	50		03 D0 00065	MOVL	#3, R0		1578
			04 00068	RET			
	3C		54 91 00069 5\$:	CMPB	R4, #60		1580
			03 12 0006C	BNEQ	6\$		
	51		03 D0 0006E	MOVL	#3, OP		1581
	3D		54 91 00071 6\$:	CMPB	R4, #61		1582
			02 12 00074	BNEQ	7\$		
			51 D4 00076	CLRL	OP		1583
	3E		54 91 00078 7\$:	CMPB	R4, #62		1584
			03 12 0007B	BNEQ	8\$		
	51		01 D0 0007D	MOVL	#1, OP		1585
	3C		54 91 00080 8\$:	CMPB	R4, #60		1586
			11 1F 00083	BLSSU	9\$		
	3E		54 91 00085	CMPB	R4, #62		
			0C 1A 00088	BGTRU	9\$		
50	56		04 78 0008A	ASHL	#4, CHAR1, R0		1587
	50	6041	DE 0008E	MOVAL	(R0)[OP], R0		
	50		53 C0 00092	ADDL2	CHAR2, R0		
			04 00095	RET			
			50 D4 00096 9\$:	CLRL	R0		1592
			04 00098	RET			

; Routine Size: 153 bytes, Routine Base: SOR\$RO_CODE + 074F


```
1306 1593 1 GLOBAL ROUTINE GET ONE CHAR(
1307 1594 1     DESC: REF VECTOR[3],
1308 1595 1     CH2: REF VECTOR[2,BYTE] ) =
1309 1596 1
1310 1597 1 ++
1311 1598 1 Functional Description:
1312 1599 1
1313 1600 1     Get the character or double character within DESC.
1314 1601 1
1315 1602 1     This is of the form:
1316 1603 1     { char | char char }
1317 1604 1
1318 1605 1     Where char is of the form:
1319 1606 1     { <quoted-string> | %<base><digits> | <character> }
1320 1607 1
1321 1608 1 Formal Parameters:
1322 1609 1
1323 1610 1     DESC    Address of length/address for
1324 1611 1             remaining chars/address of first character of the keyword
1325 1612 1     CH2     Address of two characters.
1326 1613 1             0 = First character.
1327 1614 1             Single character or
1328 1615 1             first character of double character.
1329 1616 1             1 = Second character.
1330 1617 1             Same as first character if return value GC_SINGLE, or
1331 1618 1             second character of double character.
1332 1619 1
1333 1620 1 Implicit Inputs:
1334 1621 1
1335 1622 1     None.
1336 1623 1
1337 1624 1 Implicit Outputs:
1338 1625 1
1339 1626 1     None.
1340 1627 1
1341 1628 1 Routine Value:
1342 1629 1
1343 1630 1     Number of characters found, any more than two is considered an
1344 1631 1     error
1345 1632 1
1346 1633 1 Side Effects:
1347 1634 1
1348 1635 1 --
1349 1636 2 BEGIN
1350 1637 2
1351 1638 2 LOCAL
1352 1639 2     BASE,
1353 1640 2     STR: VECTOR[3],
1354 1641 2     VALUE,
1355 1642 2     NUM_CHARS;      ! Number of characters found
1356 1643 2 MACRO
1357 1644 2     ADVANCE (X) =
1358 1645 2     BEGIN
1359 1646 2         X[_PTR] = .X[_PTR] + 1;
1360 1647 2         X[_LEN] = .X[_LEN] - 1;
1361 1648 2     END %;
1362 1649 2
```

M
M
M


```
: 1363      1650  2      output_('get_one_char');
: 1364      1651  2
: 1365      1652  2      NUM_CHARS = 0;
: 1366      1653  2      IF SKIP_IGNORED( DESC[0] ) EQL FAIL THEN RETURN FAIL;
: 1367      1654  2
: 1368      1655  2      ! While there are more characters
: 1369      1656  2      WHILE .DESC[_LEN] GTR 0 DO
: 1370      1657  2          BEGIN
: 1371      1658  3
: 1372      1659  3          SELECTONE CH$RCHAR(.DESC[_PTR]) OF
: 1373      1660  3              SET
: 1374      1661  3              [C_SPACE, C_TAB]:          ! Skip it
: 1375      1662  3                  BEGIN
: 1376      1663  4                  ADVANCE_(DESC);
: 1377      1664  4                  END;
: 1378      1665  3
: 1379      1666  3              [C_PERCENT]:          ! We have octal, hex, or decimal
: 1380      1667  3                  BEGIN
: 1381      1668  4
: 1382      1669  4                  ADVANCE_(DESC);          ! Look at the next character
: 1383      1670  4
: 1384      1671  4                  ! Out of input?
: 1385      1672  4                  IF .DESC[_LEN] LEQ 0
: 1386      1673  4                  THEN
: 1387      1674  4                      EXITLOOP;
: 1388      1675  4
: 1389      1676  4                  ! Find out what base, lower case it
: 1390      1677  4                  BASE = LOWER_(CH$RCHAR(.DESC[_PTR]));
: 1391      1678  4
: 1392      1679  4                  ! Convert base character to number
: 1393      1680  4                  SELECTONE .BASE OF
: 1394      1681  4                      SET
: 1395      1682  4                          [C_DEC]:      BASE = 10;          ! Decimal
: 1396      1683  4                          [C_OCT]:      BASE = 8;          ! Octal
: 1397      1684  4                          [C_HEX]:      BASE = 16;         ! Hexidecimal
: 1398      1685  4                          [OTHERWISE]: RETURN FAIL;    ! Bad base
: 1399      1686  4                      TES;
: 1400      1687  4
: 1401      1688  4                  ! Move off of the base character
: 1402      1689  4                  ADVANCE_(DESC);
: 1403      1690  4
: 1404      1691  4                  ! Get the number in character string form
: 1405      1692  4                  STR[_LEN] = 0;
: 1406      1693  4                  STR[_PTR] = .DESC[_PTR];
: 1407      1694  4                  WHILE .DESC[_LEN] GTR 0 DO
: 1408      1695  4                      BEGIN
: 1409      1696  5                          SELECTONE CH$RCHAR(.DESC[_PTR]) OF
: 1410      1697  5                              SET
: 1411      1698  5                                  [X'C'0' TO X'C'9', X'C'A' TO X'C'F', X'C'a' TO X'C'f']: 0;
: 1412      1699  5                                  [OTHERWISE]: EXITLOOP;
: 1413      1700  5                                  TES;
: 1414      1701  5                                  STR[_LEN] = .STR[_LEN] + 1;
: 1415      1702  5                                  ADVANCE_(DESC);
: 1416      1703  5                                  END;
: 1417      1704  4
: 1418      1705  4                  IF .STR[_LEN] EQL 0 THEN RETURN FAIL;
: 1419      1706  4
```



```

: 1420      1707  4      ! Get the number
: 1421      1708  4      IF CVT_ATB( .BASE, STR[0], VALUE ) EQL FAIL THEN RETURN FAIL;
: 1422      1709  4
: 1423      1710  4      IF .VALUE GTRU 255 THEN RETURN FAIL;
: 1424      1711  4
: 1425      1712  4      ! Put value in appropriate place(s)
: 1426      1713  4      IF .NUM_CHARS LSS 1 THEN CH2[0] = .VALUE;
: 1427      1714  4      CH2[1] = .VALUE;
: 1428      1715  4      NUM_CHARS = .NUM_CHARS + 1;
: 1429      1716  4
: 1430      1717  3      END;
: 1431      1718  3
: 1432      1719  3      [C_QUOTE]:      ! Quoted string
: 1433      1720  4      BEGIN
: 1434      1721  4      LOCAL
: 1435      1722  4      Q_CNT;
: 1436      1723  4
: 1437      1724  4      Q_CNT = 1;      ! We have seen one quote
: 1438      1725  4      ADVANCE_(DESC);      ! Advance past the first quote
: 1439      1726  4
: 1440      1727  4      ! Keep scanning characters until we are through with the string
: 1441      1728  4      !
: 1442      1729  4      WHILE TRUE DO
: 1443      1730  5      BEGIN
: 1444      1731  5
: 1445      1732  5      IF CH$RCHAR(.DESC[_PTR]) EQL C_QUOTE
: 1446      1733  5      THEN
: 1447      1734  5      Q_CNT = .Q_CNT + 1
: 1448      1735  5      ELSE
: 1449      1736  5      IF NOT .Q_CNT THEN EXITLOOP;
: 1450      1737  5
: 1451      1738  5      IF .Q_CNT
: 1452      1739  5      THEN
: 1453      1740  6      BEGIN
: 1454      1741  6      !
: 1455      1742  6      ! Emit the current character
: 1456      1743  6      !
: 1457      1744  6      CH2[1] = CH$RCHAR(.DESC[_PTR]);
: 1458      1745  6      IF .NUM_CHARS EQL 0 THEN CH2[0] = .CH2[1];
: 1459      1746  6      NUM_CHARS = .NUM_CHARS + 1;
: 1460      1747  5      END;
: 1461      1748  5
: 1462      1749  5      ADVANCE_(DESC);
: 1463      1750  4      END;
: 1464      1751  4
: 1465      1752  3      END;
: 1466      1753  3
: 1467      1754  3      [C_EQUAL, C_GREATER, C_LESS, C_DASH, C_L_PAREN, C_R_PAREN,
: 1468      1755  3      C_[F, C_CR, C_NULL, C_COMMA, C_EXCLAM]:
: 1469      1756  3      ! Reserved character so must be done
: 1470      1757  3      EXITLOOP;
: 1471      1758  3
: 1472      C 1759  3      !
: 1473      C 1760  3      [XC'A' TO XC'Z', XC'a' TO XC'z']:      ! A valid unquoted character
: 1474      C 1761  3      BEGIN
: 1475      C 1762  3      CH2[1] = CH$RCHAR(.DESC[_PTR]);
: 1476      C 1763  3      IF .NUM_CHARS EQL 0 THEN CH2[0] = .CH2[1];

```



```
: 1477      C 1764 3
: 1478      C 1765
: 1479      C 1766
: 1480      1767 )%
: 1481      1768
: 1482      1769
: 1483      1770
: 1484      1771
: 1485      1772
: 1486      1773
: 1487      1774
: 1488      1775
: 1489      1776
: 1490      1777
: 1491      1778
: 1492      1779
: 1493      1780
: 1494      1781
: 1495      1782 1

      NUM_CHARS = .NUM_CHARS + 1;
      ADVANCE_(DESC);
      END;

      [OTHERWISE]:
      RETURN FAIL;

      TES;

      END;

      ! Done
      IF .NUM_CHARS GTR 2
      THEN
      RETURN FAIL;

      RETURN .NUM_CHARS

      END;
```

			003C 00000	.ENTRY	GOC\$GET_ONE_CHAR, Save R2,R3,R4,R5	: 1593
5E		10	C2 00002	SUBL2	#16, SP	
		55	D4 00005	CLRL	NUM_CHARS	: 1652
53	04	AC	D0 00007	MOVL	DESC, R3	: 1653
		53	DD 0000B	PUSHL	R3	
FDFC	CF	01	FB 0000D	CALLS	#1, SI\$SKIP_IGNORED	
		50	D5 00012	TSTL	R0	
		03	12 00014	BNEQ	2\$	
		013D	31 00016	BRW	32\$	
52	04	A3	9E 00019	MOVAB	4(R3), R2	: 1659
		63	D5 0001D	TSTL	(R3)	: 1656
		20	15 0001F	BLEQ	7\$	
50	00	B2	9A 00021	MOVZBL	@0(R2), R0	: 1659
09		50	91 00025	CMPB	R0, #9	: 1662
		05	13 00028	BEQL	4\$	
20		50	91 0002A	CMPB	R0, #32	
		06	12 0002D	BNEQ	5\$	
		62	D6 0002F	INCL	(R2)	: 1664
		63	D7 00031	DECL	(R3)	
		E8	11 00033	BRB	3\$	: 1659
25		50	91 00035	CMPB	R0, #37	: 1667
		03	13 00038	BEQL	6\$	
		00AB	31 0003A	BRW	21\$	
		62	D6 0003D	INCL	(R2)	: 1670
		63	D7 0003F	DECL	(R3)	
		03	14 00041	BGTR	8\$	: 1673
		0107	31 00043	BRW	31\$	
54	00	B2	9A 00046	MOVZBL	@0(R2), BASE	: 1678
54		20	C8 0004A	BISL2	#32, BASE	
00000064	8F	54	D1 0004D	CMPL	BASE, #100	: 1683
		05	12 00054	BNEQ	9\$	
54		0A	D0 00056	MOVL	#10, BASE	
		1A	11 00059	BRB	11\$	

0000006F	8F		54	D1	0005B	9\$:	CMPL	BASE, #111	1684
			05	12	00062		BNEQ	10\$	
			08	D0	00064		MOVL	#8, BASE	
			0C	11	00067		BRB	11\$	
00000078	8F		54	D1	00069	10\$:	CMPL	BASE, #120	1685
			A4	12	00070		BNEQ	1\$	
			10	D0	00072		MOVL	#16, BASE	
			62	D6	00075	11\$:	INCL	(R2)	1690
			63	D7	00077		DECL	(R3)	
		04	AE	D4	00079		CLRL	STR	1693
08	AE		62	D0	0007C		MOVL	(R2), STR+4	1694
			63	D5	00080	12\$:	TSTL	(R3)	1695
			2F	15	00082		BLEQ	16\$	
		00	B2	9A	00084		MOVZBL	@0(R2) R0	1697
			50	91	00088		CMPB	R0, #48	1699
			05	1F	0008B		BLSSU	13\$	
			50	91	0008D		CMPB	R0, #57	
			18	1B	00090		BLEQU	15\$	
41	8F		50	91	00092	13\$:	CMPB	R0, #65	
			06	1F	00096		BLSSU	14\$	
46	8F		50	91	00098		CMPB	R0, #70	
			0C	1B	0009C		BLEQU	15\$	
61	8F		50	91	0009E	14\$:	CMPB	R0, #97	
			0F	1F	000A2		BLSSU	16\$	
66	8F		50	91	000A4		CMPB	R0, #102	
			09	1A	000A8		BGTRU	16\$	
		04	AE	D6	000AA	15\$:	INCL	STR	1702
			62	D6	000AD		INCL	(R2)	1703
			63	D7	000AF		DECL	(R3)	
			CD	11	000B1		BRB	12\$	1695
		04	AE	D5	000B3	16\$:	TSTL	STR	1705
			0E	13	000B6		BEQL	17\$	
			5E	DD	000B8		PUSHL	SP	1708
		08	AE	9F	000BA		PUSHAB	STR	
			54	DD	000BD		PUSHL	BASE	
FBD3	CF		03	FB	000BF		CALLS	#3, CASCVT_ATB	
			50	D5	000C4		TSTL	R0	
			03	12	000C6	17\$:	BNEQ	19\$	
		008B	31	000C8	18\$:	BRW	32\$		
000000FF	8F		6E	D1	000CB	19\$:	CMPL	VALUE, #255	1710
			F4	1A	000D2		BGTRU	18\$	
			55	D5	000D4		TSTL	NUM_CHARS	1713
			04	14	000D6		BGTR	20\$	
08	BC		6E	90	000D8		MOVB	VALUE, @CH2	
		08	AC	D0	000DC	20\$:	MOVL	CH2, R0	1714
01	A0		6E	90	000E0		MOVB	VALUE, 1(R0)	
			55	D6	000E4		INCL	NUM_CHARS	1715
			19	11	000E6		BRB	24\$	1659
			50	91	000E8	21\$:	CMPB	R0, #34	1719
			2F	12	000EB		BNEQ	28\$	
			01	D0	000ED		MOVL	#1, Q_CNT	1724
			62	D6	000F0	22\$:	INCL	(R2)	1725
			63	D7	000F2		DECL	(R3)	
		00	B2	91	000F4		CMPB	@0(R2), #34	1732
			04	12	000F8		BNEQ	23\$	
			51	D6	000FA		INCL	Q_CNT	1734
			06	11	000FC		BRB	25\$	

06		51	E8	000FE	23\$:	BLBS	Q CNT, 26\$	:	1736
		FF19	31	00101	24\$:	BRW	3\$	:	
E9		51	E9	00104	25\$:	BLBC	Q CNT, 22\$	:	1738
50	08	AC	D0	00107	26\$:	MOVL	CR2, R0	:	1744
01	00	B2	90	0010B		MOVB	20(R2), 1(R0)	:	
		55	D5	00110		TSTL	NUM_CHARS	:	1745
		04	12	00112		BNEQ	27\$	:	
60	01	A0	90	00114		MOVB	1(R0), (R0)	:	
		55	D6	00118	27\$:	INCL	NUM_CHARS	:	1746
		D4	11	0011A		BRB	22\$	:	1749
		50	D5	0011C	28\$:	TSTL	R0	:	1754
		2D	13	0011E		BEQL	31\$	:	
0A		50	91	00120		CMPB	R0, #10	:	
		28	13	00123		BEQL	31\$	:	
0D		50	91	00125		CMPB	R0, #13	:	
		23	13	00128		BEQL	31\$	:	
21		50	91	0012A		CMPB	R0, #33	:	
		1E	13	0012D		BEQL	31\$	:	
28		50	91	0012F		CMPB	R0, #40	:	
		05	1F	00132		BLSSU	29\$	:	
29		50	91	00134		CMPB	R0, #41	:	
		14	1B	00137		BLEQU	31\$	:	
2C		50	91	00139	29\$:	CMPB	R0, #44	:	
		05	1F	0013C		BLSSU	30\$	:	
2D		50	91	0013E		CMPB	R0, #45	:	
		0A	1B	00141		BLEQU	31\$	:	
3C		50	91	00143	30\$:	CMPB	R0, #60	:	
		0E	1F	00146		BLSSU	32\$	:	
3E		50	91	00148		CMPB	R0, #62	:	
		09	1A	0014B		BGTRU	32\$	:	
02		55	D1	0014D	31\$:	CMPL	NUM_CHARS, #2	:	1776
		04	14	00150		BGTR	32\$	:	
50		55	D0	00152		MOVL	NUM_CHARS, R0	:	1780
			04	00155		RET		:	
		50	D4	00156	32\$:	CLRL	R0	:	1782
			04	00158		RET		:	

; Routine Size: 345 bytes, Routine Base: SOR\$R0_CODE + 07E8


```
1497 1783 1 GLOBAL ROUTINE SEARCH_TABLE
1498 1784 1 (
1499 1785 1     STRING: REF VECTOR[2],
1500 1786 1     INDEX: REF VECTOR[1],
1501 1787 1     TABLE: REF SYM_TAB[],
1502 1788 1     COUNT
1503 1789 1 ) =
1504 1790 1 ++
1505 1791 1 Functional Description:
1506 1792 1
1507 1793 1     Search through a table to find a match with an input string.
1508 1794 1
1509 1795 1 Formal Parameters:
1510 1796 1
1511 1797 1     STRING - Address of length/address of the input string.
1512 1798 1     INDEX  - Address to store index of matching table entry.
1513 1799 1     TABLE - Address of the table.
1514 1800 1     COUNT  - The number of entries in the table
1515 1801 1
1516 1802 1 Implicit Inputs:
1517 1803 1
1518 1804 1 Implicit Outputs:
1519 1805 1
1520 1806 1 Routine Value:
1521 1807 1
1522 1808 1     SUCCESS
1523 1809 1     FAIL
1524 1810 1
1525 1811 1 Side Effects:
1526 1812 1
1527 1813 1     INDEX is set to the index of the matching table entry.
1528 1814 1
1529 1815 1 --
1530 1816 1
1531 1817 2 BEGIN
1532 1818 2
1533 1819 2 LOCAL
1534 1820 2     ITEM: VECTOR[2],          ! Length/address of table item
1535 1821 2     INP,                    ! Pointer to input string
1536 1822 2     MATCH;                  ! Match flag
1537 1823 2
1538 1824 2 output_('search_table');
1539 1825 2
1540 1826 2 ! If input string length is zero return fail
1541 1827 2 !
1542 1828 2 IF .STRING[_LEN] LEQ 0 THEN RETURN FAIL;
1543 1829 2
1544 1830 2 ! For each entry in table
1545 1831 2 !
1546 1832 2 DECR IN FROM .COUNT-1 TO 0 DO
1547 1833 2 BEGIN
1548 1834 2
1549 1835 2     ! Initialize pointer to start of input string
1550 1836 2     INP = .STRING[_PTR];
1551 1837 2
1552 1838 2     ! Get the address and length of table item
1553 1839 2     ITEM[_PTR] = .TABLE[.IN, SYM_NAM_ADR];
```



```
: 1554      1840      3      ITEM[_LEN] = .TABLE[.IN, SYM_NAM_LEN];
: 1555      1841      3
: 1556      1842      3      ! Initialize match flag
: 1557      1843      3      MATCH = TRUE;
: 1558      1844      3
: 1559      1845      3      ! Lengths must be equal
: 1560      1846      3      IF .ITEM[_LEN] EQL .STRING[_LEN]
: 1561      1847      3      THEN
: 1562      1848      4          BEGIN
: 1563      1849      4              ! Compare the strings
: 1564      1850      4              !
: 1565      1851      4              WHILE .INP LSSA .STRING[_PTR] + .STRING[_LEN] DO
: 1566      1852      4                  BEGIN
: 1567      1853      5                      ! Compare next characters converted to lower case
: 1568      1854      5                      IF LOWER_(CH$RCHAR(.INP)) NEQ LOWER_(CH$RCHAR(.ITEM[_PTR]))
: 1569      1855      5                      THEN
: 1570      1856      6                          BEGIN
: 1571      1857      5                              MATCH = FALSE;                ! They don't match
: 1572      1858      6                              EXITLOOP;
: 1573      1859      6                              END;
: 1574      1860      6                          ! Still have a match
: 1575      1861      5                          INP = .INP + 1;
: 1576      1862      5                          ITEM[_PTR] = .ITEM[_PTR] + 1;
: 1577      1863      5                      END;
: 1578      1864      5                  END;
: 1579      1865      5              ! Did they match?
: 1580      1866      5              IF .MATCH
: 1581      1867      4                  THEN
: 1582      1868      4                      BEGIN
: 1583      1869      4                          INDEX[0] = .IN;                ! Set index as output
: 1584      1870      4                          RETURN SUCCESS
: 1585      1871      4                      END;
: 1586      1872      5                  END;
: 1587      1873      5              END;
: 1588      1874      5          END;
: 1589      1875      4      ! Did not find match
: 1590      1876      4      RETURN FAIL
: 1591      1877      3      END;
: 1592      1878      3
: 1593      1879      2
: 1594      1880      2
: 1595      1881      2
: 1596      1882      2
: 1597      1883      2
: 1598      1884      1      END;
```

			007C 00000	.ENTRY	ST\$SEARCH_TABLE, Save R2,R3,R4,R5,R6	: 1783
5E		08	C2 00002	SUBL2	#8, SP	: 1828
55	04	BC	D0 00005	MOVL	@STRING, R5	: 1836
		5B	15 00009	BLEQ	6\$	: 1839
53	04	AC	D0 0000B	MOVL	STRING, R3	:
50	10	AC	D0 0000F	MOVL	COUNT, IN	:
		4E	11 00013	BRB	5\$	:

52		51	04	A3	D0	00015	1\$:	MOVL	4(R3), INP	:	1836
		50		06	C5	00019		MULL3	#6, IN, R2	:	1839
		52	0C	AC	C0	0001D		ADDL2	TABLE, R2	:	
	04	AE		62	D0	00021		MOVL	(R2), ITEM+4	:	
		6E	04	A2	9A	00025		MOVZBL	4(R2), ITEM	:	1840
		56		01	D0	00029		MOVL	#1, MATCH	:	1843
		55		6E	D1	0002C		CMPL	ITEM, R5	:	1846
				32	12	0002F		BNEQ	5\$	:	
52		55	04	A3	C1	00031	2\$:	ADDL3	4(R3), R5, R2	:	1852
		52		51	D1	00036		CMPL	INP, R2	:	
				1D	1E	00039		BGEQU	4\$	:	
		54		61	9A	0003B		MOVZBL	(INP), R4	:	1856
		54		20	C8	0003E		BISL2	#32, R4	:	
		52	04	BE	9A	00041		MOVZBL	@ITEM+4, R2	:	
		52		20	C8	00045		BISL2	#32, R2	:	
		52		54	D1	00048		CMPL	R4, R2	:	
				04	13	0004B		BEQL	3\$	:	
				56	D4	0004D		CLRL	MATCH	:	1859
				07	11	0004F		BRB	4\$	:	1858
				51	D6	00051	3\$:	INCL	INP	:	1864
			04	AE	D6	00053		INCL	ITEM+4	:	1865
				D9	11	00056		BRB	2\$	:	1852
	08	08		56	E9	00058	4\$:	BLBC	MATCH, 5\$	:	1870
	BC	50		50	D0	0005B		MOVL	IN, @INDEX	:	1873
	50			01	D0	0005F		MOVL	#1, R0	:	1874
					04	00062		RET		:	
	AF			50	F4	00063	5\$:	SOBGEQ	IN, 1\$	:	1832
				50	D4	00066	6\$:	CLRL	R0	:	1884
				04	00068			RET		:	

; Routine Size: 105 bytes, Routine Base: SOR\$RO_CODE + 0941


```
: 1600      1885 1 GLOBAL ROUTINE ERR_CNTL(
: 1601      1886 1     LINE_NO,
: 1602      1887 1     ERROR_CODE ) : CA_LINKAGE =
: 1603      1888 1
: 1604      1889 1 ++
: 1605      1890 1   Functional Description:
: 1606      1891 1       Error control routine
: 1607      1892 1
: 1608      1893 1   Formal Parameters:
: 1609      1894 1
: 1610      1895 1       LINE_NO      - Current line number.
: 1611      1896 1       ERROR_CODE   - Code number for the error being reported.
: 1612      1897 1
: 1613      1898 1   Implicit Inputs:
: 1614      1899 1
: 1615      1900 1   Implicit Outputs:
: 1616      1901 1
: 1617      1902 1   Routine Value:
: 1618      1903 1
: 1619      1904 1       SUCCESS
: 1620      1905 1       FAIL
: 1621      1906 1
: 1622      1907 1   Side Effects:
: 1623      1908 1
: 1624      1909 1 --
: 1625      1910 1
: 1626      1911 2   BEGIN
: 1627      1912 2
: 1628      1913 2   output_('err_cntl');
: 1629      1914 2
: 1630      1915 2   CA_AREA_( CA );
: 1631      1916 2
: 1632      U 1917 2   %IF %BLISS( BLISS16 ) %THEN
: 1633      U 1918 2   BEGIN
: 1634      U 1919 2
: 1635      U 1920 2   LOCAL
: 1636      U 1921 2       X,
: 1637      U 1922 2       STATUS,
: 1638      U 1923 2       ERR_BUFF : REF VECTOR [,WORD];
: 1639      U 1924 2
: 1640      U 1925 2   ! Get the address of the user error buffer
: 1641      U 1926 2   ERR_BUFF = .CA[CA_STAT_ADR];
: 1642      U 1927 2
: 1643      U 1928 2   ! Put the error code in the first word of the error buffer
: 1644      U 1929 2   ! Put the line number in the second word of the error buffer
: 1645      U 1930 2   ! Clear the rest
: 1646      U 1931 2   ERR_BUFF[0] = .ERROR_CODE;
: 1647      U 1932 2   ERR_BUFF[1] = .LINE_NO;
: 1648      U 1933 2   ERR_BUFF[2] = 0;
: 1649      U 1934 2   ERR_BUFF[3] = 0;
: 1650      U 1935 2
: 1651      U 1936 2   X = ABS( .ERROR_CODE );
: 1652      U 1937 2
: 1653      U 1938 2   ! If too big for the warning message handler then must fatal out
: 1654      U 1939 2   IF .X GEQ SOR$_LSTWRN
: 1655      U 1940 2   THEN
: 1656      U 1941 2       BEGIN
```



```
: 1657      U 1942      2      ERR_BUFF[0] = -.X;
: 1658      U 1943      2      RETURN FAIL;
: 1659      U 1944      2      END;
: 1660      U 1945      2
: 1661      U 1946      2      ! Call the user warning routine
: 1662      U 1947      2      LIB_LINK( .CA[CA_USR_WRN], ERR_BUFF[0], STATUS );
: 1663      U 1948      2
: 1664      U 1949      2      ! If we succeed then return
: 1665      U 1950      2      !
: 1666      U 1951      2      IF .STATUS NEQ SUCCESS
: 1667      U 1952      2      THEN
: 1668      U 1953      2          !warning routine may have stuffed error buffer
: 1669      U 1954      2          RETURN FAIL;
: 1670      U 1955      2
: 1671      U 1956      2      !if we haven't seen a fatal error yet then store the info away
: 1672      U 1957      2      IF .CA[ CA_1ST_SPC_ERR ] EQL 0
: 1673      U 1958      2          AND
: 1674      U 1959      2          .ERROR_CODE LSS 0
: 1675      U 1960      2      THEN
: 1676      U 1961      2          BEGIN
: 1677      U 1962      2              CA[ CA_1ST_SPC_ERR ] = .ERROR_CODE;
: 1678      U 1963      2              CA[ CA_1ST_SPC_LIN ] = .LINE_NO;
: 1679      U 1964      2          END;
: 1680      U 1965      2
: 1681      U 1966      2      !indicate no more error
: 1682      U 1967      2      ERR_BUFF[0] = 0;
: 1683      U 1968      2      ERR_BUFF[1] = 0;
: 1684      U 1969      2
: 1685      U 1970      2      END;
: 1686      U 1971      2      %FI
: 1687      U 1972      2
: 1688      L 1973      2      %IF %BLISS(BLISS32)
: 1689      U 1974      2      %THEN
: 1690      U 1975      2          BEGIN
: 1691      U 1976      2              EXTERNAL ROUTINE SOR$$ERROR;
: 1692      U 1977      2              SOR$$ERROR( .ERROR_CODE, 1, .LINE_NO );
: 1693      U 1978      2          END;
: 1694      U 1979      2      %FI
: 1695      U 1980      2
: 1696      U 1981      2      RETURN SUCCESS
: 1697      U 1982      2
: 1698      U 1983      1      END;
```

```
00000000G 00
           50
           04 AC DD 00002
           01 DD 00005
           08 AC DD 00007
           03 FB 0000A
           01 DD 00011
           04 00014
```

.EXTRN SOR\$\$ERROR

```
.ENTRY EC$ERR_CNTL, Save nothing
PUSHL LINE_NO
PUSHL 51
PUSHL ERROR_CODE
CALLS #3, SOR$$ERROR
MOVL #1, R0
RET
```

```
: 1885
: 1977
:
:
: 1981
: 1983
```

; Routine Size: 21 bytes, Routine Base: SOR\$RO_CODE + 09AA

SRTSPC
V04-000

G 2
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 55
(18)

P2
TO


```
: 1700 C 1984 1 %(
: 1701 C 1985 1 GLOBAL ROUTINE WRN_CNTL(
: 1702 C 1986 1     LINE_NO,
: 1703 C 1987 1     ERROR_CODE ): CA_LINKAGE =
: 1704 C 1988 1 ++
: 1705 C 1989 1     Functional Description:
: 1706 C 1990 1         Warning control routine
: 1707 C 1991 1
: 1708 C 1992 1     Formal Parameters:
: 1709 C 1993 1
: 1710 C 1994 1         LINE_NO           - Current line number.
: 1711 C 1995 1         ERROR_CODE        - Code number for the warning being reported.
: 1712 C 1996 1
: 1713 C 1997 1     Implicit Inputs:
: 1714 C 1998 1
: 1715 C 1999 1     Implicit Outputs:
: 1716 C 2000 1
: 1717 C 2001 1     Routine Value:
: 1718 C 2002 1
: 1719 C 2003 1         SUCCESS
: 1720 C 2004 1         FAIL
: 1721 C 2005 1
: 1722 C 2006 1     Side Effects:
: 1723 C 2007 1
: 1724 C 2008 1 --
: 1725 C 2009 1
: 1726 C 2010 1     BEGIN
: 1727 C 2011 1
: 1728 C 2012 1     LOCAL
: 1729 C 2013 1     STATUS;
: 1730 C 2014 1
: 1731 C 2015 1     ! Get the impure data area
: 1732 C 2016 1     CA_AREA_( CA );
: 1733 C 2017 1
: 1734 C 2018 1     output_('wrn_cntl');
: 1735 C 2019 1
: 1736 C 2020 1     %IF %BLISS(BLISS16)
: 1737 C 2021 1     %THEN
: 1738 C 2022 1         BEGIN
: 1739 C 2023 1             LOCAL
: 1740 C 2024 1             ERR_BUFF:  REF VECTOR [,WORD];
: 1741 C 2025 1
: 1742 C 2026 1             ! Get the address of the user error buffer
: 1743 C 2027 1             ERR_BUFF = .CA[CA_STAT_ADR];
: 1744 C 2028 1
: 1745 C 2029 1             ! Put the error code in the first word of the error buffer
: 1746 C 2030 1             ! Put the line number in the second word of the error buffer
: 1747 C 2031 1             ! Clear the third word
: 1748 C 2032 1             ERR_BUFF[0] = .ERROR_CODE;
: 1749 C 2033 1             ERR_BUFF[1] = .LINE_NO;
: 1750 C 2034 1             ERR_BUFF[2] = 0;
: 1751 C 2035 1
: 1752 C 2036 1             ! Call the user warning routine
: 1753 C 2037 1             LIB_LINK( .CA[CA_USR_WRN], ERR_BUFF[0], STATUS );
: 1754 C 2038 1
: 1755 C 2039 1             ! If we fail then put negative code in buffer
: 1756 C 2040 1             IF .STATUS NEQ SUCCESS
```



```
: 1757      C 2041 1      THEN
: 1758      C 2042 1      BEGIN
: 1759      C 2043 1
: 1760      C 2044 1      ERR_BUFF[0] = - ABS( .ERROR_CODE );
: 1761      C 2045 1
: 1762      C 2046 1      RETURN FAIL;
: 1763      C 2047 1
: 1764      C 2048 1      END;
: 1765      C 2049 1      END;
: 1766      C 2050 1      %FI
: 1767      C 2051 1      %IF %BLISS(BLISS32)
: 1768      C 2052 1      %THEN
: 1769      C 2053 1      BEGIN
: 1770      C 2054 1      EXTERNAL ROUTINE SOR$$ERROR;
: 1771      C 2055 1      SOR$$ERROR(.ERROR_CODE, 1, .LINE_NO);
: 1772      C 2056 1      END;
: 1773      C 2057 1      %FI
: 1774      C 2058 1
: 1775      C 2059 1      RETURN SUCCESS
: 1776      C 2060 1
: 1777      C 2061 1      END;
: 1778      C 2062 1      )%
: 1779      C 2063 1
: 1780      C 2064 1      ! ELUDOM
: 1781      C 2065 1      !
: 1782      C 2066 1      END
: 1783      C 2067 0      ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
SOR\$RO_CODE	2495	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	104	25	34	00:00.1
-\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	36	30	12	00:00.1
-\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI
)

```
: 1784      L 2068 0 MODULE P1$PAS  (IDENT = 'T03-038' %REQUIRE('SRC$:DEFS0') ) =
: 1785      2074 1 BEGIN
: 1786      2075 1 REQUIRE 'SRC$:DKS';
: 1787      2373 1 EXTERNAL ROUTINE
: 1788      2374 1      ERR_CNTL:      CA_LINKAGE,      ! Error control routine
: 1789      2375 1      SPC_ALLOC:    CA_LINKAGE,      ! Allocate storage from the work area
: 1790      2376 1      SKIP_IGNORED,      ! Skip ignored text characters
: 1791      2377 1      SKIP_COMMA,      ! Skip commas and ignored stuff
: 1792      2378 1      GET_RYW_TYPE,      ! Get keyword type
: 1793      2379 1      GET_CHAR_CLAUSE,      ! Get character clauses of various kinds
: 1794      2380 1      GET_ONE_CHAR,      ! Get character or double character
: 1795      2381 1      GET_NEXT_SPEC,      ! Get pointer and length of next specification
: 1796      2382 1      GET_SUB_SPEC,      ! Get pointer and length of sub-specification
: 1797      2383 1      GET_FILE_SPEC,      ! Find file specification
: 1798      2384 1      PARSE_COLL:    CA_LINKAGE;      ! Parse collating sequence
```



```
1800 2385 1 GLOBAL ROUTINE P1$PASS_ONE(
1801 2386 1     START:      REF VECTOR[3],
1802 2387 1     CON_SYM:   REF CON_SYM TAB,
1803 2388 1     P_CON_CNT:  REF VECTOR[1]
1804 2389 1 );      CA_LINKAGE =
1805 2390 2 BEGIN
1806 2391 2 -----
1807 2392 2 Pass 1:  parse /process, /stable, /strip,
1808 2393 2         and /collating_sequence specifications.
1809 2394 2         Build the PT and ST tables
1810 2395 2 -----
1811 2396 2 BIND
1812 2397 2     CON_CNT = P_CON_CNT[0];
1813 2398 2
1814 2399 2 LOCAL
1815 2400 2     KYW_LINE,           ! Line number of keyword
1816 2401 2     KYW_TYPE,           ! Keyword type
1817 2402 2     SUB_KYW_TYPE,
1818 2403 2     BUFF:      VECTOR[3],
1819 2404 2     SPC:      VECTOR[3],      ! Next specification length/pointer/line
1820 2405 2     SUB_SPC:  VECTOR[3];    ! Sub specification length/pointer/line
1821 2406 2
1822 2407 2 ! Get the impure area
1823 2408 2 CA_AREA_( CA );
1824 2409 2
1825 2410 2 -----
1826 2411 2 Pass 1:  parse /process, /stable, /strip,
1827 2412 2         and /collating_sequence specifications.
1828 2413 2         Build the PT and ST tables
1829 2414 2 -----
1830 2415 3 BEGIN                                ! Pass 1
1831 2416 3 LOCAL
1832 2417 3     COLL:  VECTOR[3];                ! Collating sequence specification
1833 2418 3
1834 2419 3 output_('pass 1');
1835 2420 3
1836 2421 3 ! Initialize for pass 1
1837 2422 3 BUFF[_LEN] = .START[_LEN];
1838 2423 3 BUFF[_PTR] = .START[_PTR];
1839 2424 3 BUFF[_LINE] = .START[_LINE];
1840 2425 3
1841 2426 3 COLL[_LEN] = 0;                        ! No collating sequence yet
1842 2427 3
1843 2428 3 ! Exit loop when there are no remaining specifications
1844 2429 3
1845 2430 3 WHILE TRUE DO
1846 2431 4 BEGIN
1847 2432 4
1848 2433 4     KYW_LINE = .BUFF[_LINE];
1849 2434 4     KYW_TYPE = GET_NEXT_SPEC( BUFF[0], SPC[0] );
1850 2435 4     SUB_SPC[_LEN] = 0;                ! Simplifies error checking below
1851 2436 4
1852 2437 4
1853 2438 4     ! Case on the keyword
1854 2439 4     !
1855 2440 4     SELECTONE .KYW_TYPE OF
1856 2441 4         SET
```


1857 2442 4
1858 2443 4
1859 2444 5
1860 2445 5
1861 2446 5
1862 2447 6
1863 2448 5
1864 2449 5
1865 2450 6
1866 2451 5
1867 2452 6
1868 2453 6
1869 2454 6
1870 2455 6
1871 2456 6
1872 2457 6
1873 2458 6
1874 2459 6
1875 2460 6
1876 2461 6
1877 2462 6
1878 2463 6
1879 2464 6
1880 2465 6
1881 2466 5
1882 2467 5
1883 2468 4
1884 2469 4
1885 2470 4
1886 2471 5
1887 2472 5
1888 2473 5
1889 2474 6
1890 2475 5
1891 2476 6
1892 2477 6
1893 2478 6
1894 2479 6
1895 2480 5
1896 2481 4
1897 2482 4
1898 2483 4
1899 2484 5
1900 2485 5
1901 2486 5
1902 2487 6
1903 2488 5
1904 2489 5
1905 2490 6
1906 2491 5
1907 2492 6
1908 2493 6
1909 2494 6
1910 2495 6
1911 2496 5
1912 2497 4
1913 2498 4

```
[KW_PROC]:                                ! /PROCESS=
BEGIN
  IF .CA[CA_VAR_MERGE]                    ! Sort only
  THEN
    WRN_( SORS_SPCMIS )                   ! Merge invalid specification
  ELSE IF .CA[CA_PROCESS_OVR]
  THEN
    WRN_( SORS_SPCOVR )                   ! Warn of override
  ELSE
    BEGIN
      KYW_LINE = .SPC[ LINE];             ! Save line for errors
      SUB_KYW_TYPE = GET_SUB_SPEC( SPC[0], SUB_SPC[0] );
      SELECTONE .SUB_KYW_TYPE OF
        SET
          [KW_REC'D]:    CA[CA_PROCESS] = TYP_K_RECORD;
          [KW_TAG]:      CA[CA_PROCESS] = TYP_K_TAG;
          [KW_ADDR]:     CA[CA_PROCESS] = TYP_K_ADDRESS;
          [KW_INDX]:     CA[CA_PROCESS] = TYP_K_INDEX;
          [OTHERWISE]:   ERR_( .KYW_LINE, SORS_SPCIVP );
        TES;
      END;
      SPC[ _LEN ] = 0;
    END;
  [KW_STAB, KW_NOST]:                      ! /[NO]STABLE
  BEGIN
    IF .CA[CA_STABLE_OVR]
    THEN
      WRN_( SORS_SPCOVR )                 ! Warn of override
    ELSE
      BEGIN
        IF .KYW_TYPE EQL KW_STAB
        THEN CA[CA_STAB[E]] = TRUE
        ELSE CA[CA_STABLE] = FALSE;
      END;
    END;
  [KW_CHSQ, KW_NOCH]:                      ! /[NO]CHECK_SEQUENCE
  BEGIN
    IF NOT .CA[CA_VAR_MERGE]              ! Merge only
    THEN
      WRN_( SORS_SPCIS )                  ! Sort invalid specification
    ELSE IF .CA[CA_CHKSEQ_OVR]
    THEN
      WRN_( SORS_SPCOVR )                 ! Warn of override
    ELSE
      BEGIN
        IF .KYW_TYPE EQL KW_CHSQ
        THEN CA[CA_CHKSEQ] = TRUE
        ELSE CA[CA_CHKSEQ] = FALSE
      END;
    END;
  END;
```


1914 2499 4
1915 2500 5
1916 2501 5
1917 2502 5
1918 2503 6
1919 2504 5
1920 2505 6
1921 2506 6
1922 2507 6
1923 2508 6
1924 2509 5
1925 2510 5
1926 2511 4
1927 2512 4
1928 2513 4
1929 2514 5
1930 2515 5
1931 2516 5
1932 2517 5
1933 2518 5
1934 2519 6
1935 2520 5
1936 2521 5
1937 2522 4
1938 2523 4
1939 2524 4
1940 2525 4
1941 2526 5
1942 2527 5
1943 2528 4
1944 2529 4
1945 2530 4
1946 2531 5
1947 2532 5
1948 2533 5
1949 2534 5
1950 2535 5
1951 2536 6
1952 2537 6
1953 2538 6
1954 2539 6
1955 2540 6
1956 2541 6
1957 2542 6
1958 2543 6
1959 2544 6
1960 2545 6
1961 2546 5
1962 2547 5
1963 2548 4
1964 2549 4
1965 2550 4
1966 2551 5
1967 2552 5
1968 2553 5
1969 2554 4
1970 2555 4

```
[KW_COLL]:
BEGIN
  IF .CA[CA_COLSEQ_OVR]
  THEN
    WRN_( SOR$_SPCOVR )      ! Warn of override
  ELSE
    BEGIN
      COLL[_LEN] = .SPC[_LEN];
      COLL[_PTR] = .SPC[_PTR];
      COLL[_LINE] = .SPC[_LINE];
    END;
    SPC[_LEN] = 0;
  END;

[KW_PAD]:
BEGIN
  LOCAL
    CH2: VECTOR[2,BYTE];
  IF GET_ONE_CHAR(SPC[0], CH2) NEQ 1
  THEN
    ERR_( .KYW_LINE, SOR$_SPCIVS )      ! Invalid specification
  ELSE
    CA[CA_PAD] = .CH2[0];
  END;

[KW_FIEL, KW_INCL, KW_OMIT, KW_KEY, KW_DATA,
KW_COND, KW_CDD]:
BEGIN
  SPC[_LEN] = 0;      ! Not parsed this pass
END;

[KW_WORK]:
BEGIN
  LOCAL
    P: REF VECTOR[BYTE];
  P = CA[CA_WF_NAMES];
  WHILE TRUE DO
    BEGIN
      IF GET_FILE_SPEC( SPC[0], SUB_SPC[0] ) EQL FAIL THEN EXITLOOP;
      IF .P[0] GEQU MAX_WORK_FILES THEN EXITLOOP;
      P[0] = .P[0] + 1;
      P[.P[0]] = .CON_CNT;
      CON_SYM[.CON_CNT, CON_NAM_ADR] = .SUB_SPC[_PTR];
      CON_SYM[.CON_CNT, CON_NAM_LEN] = .SUB_SPC[_LEN];
      CON_SYM[.CON_CNT, CON_LENGTH] = .SUB_SPC[_LEN];
      CON_CNT = .CON_CNT + 1;
      IF SKIP_COMMA(SPC[0]) EQL FAIL THEN EXITLOOP;
    END;
    SUB_SPC[_LEN] = 0;
  END;

[OTHERWISE]:      ! An invalid specification keyword
BEGIN
  ERR_( .KYW_LINE, SOR$_SPCIVS ); ! Invalid specification
  SPC[_LEN] = 0;      ! Not parsed this pass
END;
```



```

: 1971      2556  4      TES;
: 1972      2557  4
: 1973      2558  4      ! Check for extraneous text
: 1974      2559  4
: 1975      2560  4      IF .SUB_SPC[_LEN] NEQ 0      ! Extraneous text?
: 1976      2561  4      THEN
: 1977      2562  4          ERR ( .KYW_LINE, SOR$_SPCIVS );
: 1978      2563  4      IF .SPC[_LEN] NEQ 0      ! Extraneous text?
: 1979      2564  4      THEN
: 1980      2565  4          ERR ( .SPC[_LINE], SOR$_SPCIVS );      ! Invalid specification
: 1981      2566  4
: 1982      2567  4      ! Look for next significant character
: 1983      2568  4      IF SKIP_IGNORED( BUFF[0] ) EQL FAIL THEN EXITLOOP;
: 1984      2569  4
: 1985      2570  4      END;
: 1986      2571  4
: 1987      2572  4      ! Process the collating sequence (now that we know the pad character)
: 1988      2573  4
: 1989      2574  4      IF .COLL[_LEN] NEQ 0
: 1990      2575  4      THEN
: 1991      2576  4          IF PARSE_COLL( COLL[0] ) EQL FAIL THEN RETURN FAIL;
: 1992      2577  4
: 1993      2578  4      END;
: 1994      2579  4      ! Pass 1
: 1995      2580  4
: 1996      2581  4      RETURN SUCCESS;
: 1997      2582  4      END;
```

```
.TITLE P1$PAS
.IDENT \T03-038\
```

```
.EXTRN EC$ERR_CNTL, SASSPC_ALLOC
.EXTRN SI$SKIP_IGNORED
.EXTRN SC$SKIP_COMMA, GKT$GET_KYW_TYPE
.EXTRN GCC$GET_CHAR_CLAUSE
.EXTRN GOC$GET_ONE_CHAR
.EXTRN GNS$GET_NEXT_SPEC
.EXTRN GSS$GET_SUB_SPEC
.EXTRN GFS$GET_FILE_SPEC
.EXTRN PC$PARSE_COLL
```

```
.PSECT SOR$RO_CODE, NOWRT, SHR, PIC, 2
```

```

                                007C 00000
                                56 00000000G 00 9E 00002
                                5E          34 C2 00009
                                50          04 AC D0 0000C
                                28 AE          60 7D 00010
                                30 AE          08 A0 D0 00014
                                54          04 AE D4 00019
                                1C AE          30 D0 0001C 1$:
                                2C AE          1C 9F 00020
                                00000000G 00 02 FB 00026
                                53          50 D0 0002D
                                10 AE          D4 00030
```

```
.ENTRY P1$PASS ONE, Save R2,R3,R4,R5,R6      : 2385
MOVAB EC$ERR_CNTL, R6
SUBL2 #52, SP
MOVL START, R0      : 2422
MOVQ (R0), BUFF
MOVL 8(R0), BUFF+8      : 2424
CLRL COLL      : 2426
MOVL BUFF+8, KYW_LINE      : 2433
PUSHAB SPC      : 2434
PUSHAB BUFF
CALLS #2, GNS$GET_NEXT_SPEC
MOVL R0, KYW_TYPE
CLRL SUB_SPC      : 2435
```


		53	D5	00033	TSTL	KYW_TYPE	: 2443
		63	12	00035	BNEQ	10\$	:
	5C	AB	95	00037	TSTB	92(CA)	: 2445
		08	18	0003A	BGEQ	2\$	:
	001C8168	8F	DD	0003C	PUSHL	#1868136	: 2447
		53	11	00042	BRB	9\$	:
	08	79	AB	E9	00044	2\$: BLBC	121(CA), 3\$: 2448
	001C8160	8F	DD	00048	PUSHL	#1868128	: 2450
		47	11	0004E	BRB	9\$	:
	54	24	AE	D0	00050	3\$: MOVL	SPC+8, KYW_LINE : 2454
		10	AE	9F	00054	PUSHAB	SUB_SPC : 2455
		20	AE	9F	00057	PUSHAB	SPC
00000000G	00	02	FB	0005A	CALLS	#2, GSS\$GET_SUB_SPEC	:
	55	50	D0	00061	MOVL	R0, SUB_KYW_TYPE	:
	01	55	D1	00064	CMPL	SUB_KYW_TYPE, #1	: 2459
		06	12	00067	BNEQ	4\$	:
58	AB	01	90	00069	MOVB	#1, 88(CA)	:
		1F	11	0006D	BRB	7\$	:
	02	55	D1	0006F	4\$: CMPL	SUB_KYW_TYPE, #2 : 2460	
		06	12	00072	BNEQ	5\$	:
58	AB	02	90	00074	MOVB	#2, 88(CA)	:
		14	11	00078	BRB	7\$	:
	03	55	D1	0007A	5\$: CMPL	SUB_KYW_TYPE, #3 : 2461	
		06	12	0007D	BNEQ	6\$	:
58	AB	04	90	0007F	MOVB	#4, 88(CA)	:
		09	11	00083	BRB	7\$	:
	04	55	D1	00085	6\$: CMPL	SUB_KYW_TYPE, #4 : 2462	
		07	12	00088	BNEQ	8\$	:
58	AB	03	90	0008A	MOVB	#3, 88(CA)	:
		013A	31	0008E	7\$: BRW	34\$: 2463	
	001C817C	8F	DD	00091	8\$: PUSHL	#1868156 : 2470	
		0127	31	00097	9\$: BRW	33\$: 2472	
	05	53	D1	0009A	10\$: CMPL	KYW_TYPE, #5 : 2474	
		22	19	0009D	BLSS	14\$	:
	06	53	D1	0009F	CMPL	KYW_TYPE, #6	:
		1D	14	000A2	BGTR	14\$	:
	08	5B	AB	E9	000A4	BLBC	91(CA), 12\$: 2477
	001C8160	8F	DD	000A8	11\$: PUSHL	#1868128 : 2478	
		26	11	000AE	BRB	15\$	:
	05	53	D1	000B0	12\$: CMPL	KYW_TYPE, #5 : 2479	
		06	12	000B3	BNEQ	13\$	:
5B	AB	01	88	000B5	BISB2	#1, 91(CA)	: 2480
		31	11	000B9	BRB	18\$	:
5B	AB	01	8A	000BB	13\$: BICB2	#1, 91(CA) : 2483	
		2B	11	000BF	BRB	18\$	:
	07	53	D1	000C1	14\$: CMPL	KYW_TYPE, #7 : 2485	
		28	19	000C4	BLSS	19\$	:
	08	53	D1	000C6	CMPL	KYW_TYPE, #8	:
		23	14	000C9	BGTR	19\$	:
	5C	AB	95	000CB	TSTB	92(CA)	: 2487
		08	19	000CE	BLSS	16\$	:
	001C8170	8F	DD	000D0	PUSHL	#1868144	: 2488
		5B	11	000D6	15\$: BRB	22\$: 2493	
CB	5B	AB	01	E0	000D8	16\$: BBS	#1, 91(CA), 11\$: 2494
		07	53	D1	000DD	CMPL	KYW_TYPE, #7
			06	12	000E0	BNEQ	17\$
	5B	AB	02	88	000E2	BISB2	#2, 91(CA)

5B	AB		5D	11	000E6	BRB	25\$		
			02	8A	000E8	BICB2	#2, 91(CA)		2495
	09		57	11	000EC	BRB	25\$		2440
			53	D1	000EE	CMPL	KYW_TYPE, #9		2499
12	79	AB	24	12	000F1	BNEQ	21\$		
		001C8160	02	E1	000F3	BBC	#2, 121(CA), 20\$		2501
			8F	DD	000F8	PUSHL	#1868128		2503
	66		54	DD	000FE	PUSHL	KYW_LINE		
	01		02	FB	00100	CALLS	#2, EC\$ERR_CNTL		
			50	D1	00103	CMPL	R0, #1		
			86	13	00106	BEQL	7\$		
			33	11	00108	BRB	23\$		
04	AE	1C	AE	7D	0010A	MOVQ	SPC, COLL		2506
OC	AE	24	AE	D0	0010F	MOVL	SPC+8, COLL+8		2508
			00B4	31	00114	BRW	34\$		2510
	3F		53	D1	00117	CMPL	KYW_TYPE, #63		2513
			2B	12	0011A	BNEQ	26\$		
			5E	DD	0011C	PUSHL	SP		2517
		20	AE	9F	0011E	PUSHAB	SPC		
00000000G	00		02	FB	00121	CALLS	#2, GOC\$GET_ONE_CHAR		
	01		50	D1	00128	CMPL	R0, #1		
		001C8184	13	13	0012B	BEQL	24\$		
			8F	DD	0012D	PUSHL	#1868164		2519
	66		54	DD	00133	PUSHL	KYW_LINE		
	01		02	FB	00135	CALLS	#2, EC\$ERR_CNTL		
			50	D1	00138	CMPL	R0, #1		
			7C	13	0013B	BEQL	31\$		
			00E1	31	0013D	BRW	40\$		
0101	CB		6E	90	00140	MOVB	CH2, 257(CA)		2521
			72	11	00145	BRB	31\$		2440
	13		53	D1	00147	CMPL	KYW_TYPE, #19		2524
			7F	13	0014A	BEQL	34\$		
	2A		53	D1	0014C	CMPL	KYW_TYPE, #42		
			05	19	0014F	BLSS	27\$		
	2C		53	D1	00151	CMPL	KYW_TYPE, #44		
			75	15	00154	BLEQ	34\$		
	32		53	D1	00156	CMPL	KYW_TYPE, #50		
			05	19	00159	BLSS	28\$		
	33		53	D1	0015B	CMPL	KYW_TYPE, #51		
			6B	15	0015E	BLEQ	34\$		
	3E		53	D1	00160	CMPL	KYW_TYPE, #62		
			66	13	00163	BEQL	34\$		
	3D		53	D1	00165	CMPL	KYW_TYPE, #61		2530
			51	12	00168	BNEQ	32\$		
	52	0134	CB	9E	0016A	MOVAB	308(R11), P		2534
		10	AE	9F	0016F	PUSHAB	SUB_SPC		2537
		20	AE	9F	00172	PUSHAB	SPC		
00000000G	00		02	FB	00175	CALLS	#2, GFS\$GET_FILE_SPEC		
			50	D5	0017C	TSTL	R0		
			36	13	0017E	BEQL	30\$		
	0A		62	91	00180	CMPB	(P), #10		2536
			31	1E	00183	BGEQU	30\$		
			62	96	00185	INCB	(P)		2539
	51		62	9A	00187	MOVZBL	(P), R1		2540
	50	OC	BC	D0	0018A	MOVL	@P_CON_CNT, R0		
6142			50	90	0018E	MOVB	R0, (RT)[P]		
50		08	BC40	7E	00192	MOVAQ	@CON_SYM[R0], R0		2541

	60	14	AE	D0	00197	MOVL	SUB_SPC+4, (R0)		
04	A0	10	AE	90	0019B	MOVB	SUB_SPC, 4(R0)	2542	
06	A0	10	AE	90	001A0	MOVB	SUB_SPC, 6(R0)	2543	
		0C	BC	D6	001A5	INCL	AP_CON_CNT	2544	
		1C	AE	9F	001A8	PUSHAB	SPC	2545	
00000000G	00		01	FB	001AB	CALLS	#1, SC\$SKIP_COMMA		
			50	D5	001B2	TSTL	R0		
		10	B9	12	001B4	BNEQ	29\$	2547	
			AE	D4	001B6	CLRL	SUB_SPC	2440	
			13	11	001B9	BRB	35\$	2552	
		001C8184	8F	DD	001BB	PUSHL	#1868164		
			54	DD	001C1	PUSHL	KYW_LINE		
			02	FB	001C3	CALLS	#2, EC\$ERR_CNTL		
66			50	D1	001C6	CMPL	R0, #1		
01			56	12	001C9	BNEQ	40\$	2553	
		1C	AE	D4	001CB	CLRL	SPC	2560	
		10	AE	D5	001CE	TSTL	SUB_SPC		
			10	13	001D1	BEQL	36\$	2562	
		001C8184	8F	DD	001D3	PUSHL	#1868164		
			54	DD	001D9	PUSHL	KYW_LINE		
			02	FB	001DB	CALLS	#2, EC\$ERR_CNTL		
66			50	D1	001DE	CMPL	R0, #1		
01			3E	12	001E1	BNEQ	40\$	2563	
		1C	AE	D5	001E3	TSTL	SPC	2565	
			11	13	001E6	BEQL	37\$		
		001C8184	8F	DD	001E8	PUSHL	#1868164		
		28	AE	DD	001EE	PUSHL	SPC+8		
			02	FB	001F1	CALLS	#2, EC\$ERR_CNTL		
66			50	D1	001F4	CMPL	R0, #1		
01			28	12	001F7	BNEQ	40\$	2569	
			AE	9F	001F9	PUSHAB	BUFF		
00000000G	00	28	01	FB	001FC	CALLS	#1, SI\$SKIP_IGNORED		
			50	D5	00203	TSTL	R0		
			03	13	00205	BEQL	38\$		
			FE12	31	00207	BRW	1\$	2575	
		04	AE	D5	0020A	TSTL	COLL	2577	
			0E	13	0020D	BEQL	39\$		
		04	AE	9F	0020F	PUSHAB	COLL		
00000000G	00		01	FB	00212	CALLS	#1, PC\$PARSE_COLL		
			50	D5	00219	TSTL	R0		
			04	13	0021B	BEQL	40\$	2581	
		50	01	D0	0021D	MOVL	#1, R0	2582	
				04	00220	RET			
			50	D4	00221	CLRL	R0		
			04	00223	RET				

; Routine Size: 548 bytes, Routine Base: SOR\$RO_CODE + 0000

: 1998	2583	1	
: 1999	2584	1	! ELUDOM
: 2000	2585	1	!
: 2001	2586	1	END
: 2002	2587	0	ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
SORSRO_CODE	548	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	112	27	34	00:00.1
-\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	16	13	12	00:00.1
-\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI)

2003	L 2588	0	MODULE P2\$PAS	(IDENT = 'T03-038' %REQUIRE('SRC\$:DEFS0')) =	
2004	2594	1	BEGIN		
2005	2595	1	REQUIRE 'SRC\$:DKS';		
2006	2893	1	EXTERNAL ROUTINE		
2007	2894	1	ERR_CNTL:	CA_LINKAGE,	! Error control routine
2008	2895	1	SEARCH_TABLE,		! Search a table for a string
2009	2896	1	CVT_ATB,		! Convert ascii to binary
2010	2897	1	SPC_ALLOC:	CA_LINKAGE,	! Allocate storage from the work area
2011	2898	1	SKIP_IGNORED,		! Skip ignored text characters
2012	2899	1	SKIP_COMMA,		! Skip commas and ignored stuff
2013	2900	1	GET_NEXT_SPEC,		! Get pointer and length of next specification
2014	2901	1	GET_SUB_SPEC;		! Get pointer and length of sub-specification


```
2016 2902 1 GLOBAL ROUTINE P2$PASS_TWO(  
2017 2903 1     START:      REF VECTOR[3],  
2018 2904 1     FLD_SYM:  REF SYM_TAB  
2019 2905 1 ):      CA_LINKAGE =  
2020 2906 2 BEGIN  
2021 2907 2 -----  
2022 2908 2     Pass 2:  define all fields, fill the field definition  
2023 2909 2             table and the field symbol table  
2024 2910 2             Build the FDT table  
2025 2911 2     -----  
2026 2912 2  
2027 2913 2 LOCAL  
2028 2914 2     KYW_TYPE,      ! Type of keyword  
2029 2915 2     SUB_KYW_TYPE,  
2030 2916 2     KYW_LINE,      ! Line number of keyword  
2031 2917 2     BUFF:      VECTOR[3],  
2032 2918 2     SPC:      VECTOR[3], ! Next specification length/pointer/line  
2033 2919 2     SUB_SPC:    VECTOR[3]; ! Sub specification length/pointer/line  
2034 2920 2  
2035 2921 2 ! Get the impure area  
2036 2922 2 CA_AREA_( CA );  
2037 2923 2  
2038 2924 2  
2039 2925 2 -----  
2040 2926 2     Pass 2:  define all fields, fill the field definition  
2041 2927 2             table and the field symbol table  
2042 2928 2             Build the FDT table  
2043 2929 2     -----  
2044 2930 2 BEGIN                                ! Pass 2  
2045 2931 2 LOCAL  
2046 2932 2     FDT:      REF FDT_TAB[]; ! Pointer to Field Defn Table  
2047 2933 2  
2048 2934 2 output_('pass 2');  
2049 2935 2  
2050 2936 2 ! Anchor field defn table at beginning of available work area  
2051 2937 2 ! Set address of field definition table in common area.  
2052 2938 2  
2053 2939 2 %IF %BLISS(BLISS16) %THEN IF NOT .CA[CA_KEY_OVR] %FI  
2054 2940 2 %IF %BLISS(BLISS32) %THEN IF .CA[CA_FDT_ADR] EQL 0 %FI  
2055 2941 2 THEN  
2056 2942 2     BEGIN  
2057 2943 2     CA[CA_FDT_ADR] = ALLOC_(0);  
2058 2944 2     CA[CA_FDT_SIZ] = 0;  
2059 2945 2     END  
2060 2946 2 ELSE  
2061 2947 2     BEGIN  
2062 2948 2     LOCAL  
2063 2949 2     FROM_PTR;  
2064 2950 2     !move the key table entries generated by command line /ke:  
2065 2951 2     FROM_PTR = .CA[CA_FDT_ADR];  
2066 2952 2     CA[CA_FDT_ADR] = ALLOC ( .CA[CA_FDT_SIZ] * FDT_UNIT );  
2067 2953 2     CH$MOVE( .CA[CA_FDT_SIZ] * FDT_UNIT, .FROM_PTR, .CA[CA_FDT_ADR] );  
2068 2954 2     END;  
2069 2955 2  
2070 2956 2 ! Initialize for pass 2  
2071 2957 2 BUFF[_LEN] = .START[_LEN];  
2072 2958 2 BUFF[_PTR] = .START[_PTR];
```



```
2073 2959 3 BUFF[_LINE] = .START[_LINE];
2074 2960 3
2075 2961 3
2076 2962 3 ! Always define a default field; character, offset 0, lrl long
2077 2963 3
2078 2964 3 FDT = ALLOC (FDT_UNIT);
2079 2965 3 CA[CA_FDT_SIZ] = -.CA[CA_FDT_SIZ] + 1;
2080 2966 3 FDT[0, FDT_TYPE] = DT_T;
2081 2967 3 FDT[0, FDT_SCALE] = 0;
2082 2968 3 FDT[0, FDT_FLD_POS] = 0;
2083 2969 3 FDT[0, FDT_FLD_SIZ] = -1; ! (Just in case)
2084 2970 3
2085 2971 3 ! Exit loop when there are no remaining specifications
2086 2972 3 WHILE TRUE DO
2087 2973 4 BEGIN
2088 2974 4
2089 2975 4 LABEL
2090 2976 4 fld;
2091 2977 4
2092 2978 4 KYW_LINE = .BUFF[_LINE];
2093 2979 4 KYW_TYPE = GET_NEXT_SPEC( BUFF[0], SPC[0] );
2094 2980 4
2095 2981 4 ! Process the keyword (only field definitions)
2096 2982 4
2097 2983 4 SELECTONE .KYW_TYPE OF
2098 2984 4 SET
2099 2985 4 [KW_FIEL]:
2100 2986 5 fld: BEGIN ! Make an entry in the field table
2101 2987 5
2102 2988 5 LITERAL
2103 2989 5 BINARY_BIT = %X'0001'; ! Bit masks for data_types
2104 2990 5 DECIMA_BIT = %X'0002'; ! Binary
2105 2991 5 SIGNED_BIT = %X'0004'; ! Decimal
2106 2992 5 UNSIGN_BIT = %X'0008'; ! Signed
2107 2993 5 TRAILI_BIT = %X'0010'; ! Unsigned
2108 2994 5 LEADIN_BIT = %X'0020'; ! Trailing sign
2109 2995 5 SEPARA_BIT = %X'0040'; ! Leading sign
2110 2996 5 OVERPU_BIT = %X'0080'; ! Separate sign
2111 2997 5 OTHERS_BIT = %X'0100'; ! Overpunched sign
2112 2998 5 ! Other explicit data types
2113 2999 5
2114 3000 5 LOCAL
2115 3001 5 DATA_TYPE, ! Bit_vector defining data type
2116 3002 5 TYP: "BYTE", SIGNED, ! Data type
2117 3003 5 NUMBER, ! Returned number
2118 3004 5 INDEX, ! Returned index
2119 3005 5 NAM_SET, ! Name set flag
2120 3006 5 POS_SET, ! Position set flag
2121 3007 5 SIZ_SET, ! Size set flag
2122 3008 5 DIG_SET; ! Digits set flag
2123 3009 5
2124 3010 5 FDT = ALLOC (FDT_UNIT);
2125 3011 5 CA[CA_FDT_SIZ] = -.CA[CA_FDT_SIZ] + 1;
2126 3012 5
2127 3013 5 ! Too many entries in FDT or field symbol table?
2128 3014 5 IF .CA[CA_FDT_SIZ] GEQ MIN(FDT_MAX, MAX_FIELDS)
2129 3015 6 THEN
BEGIN
```



```
2130      3016 6      ERR ( .KYW_LINE, SOR$_SPCIVF );      ! Invalid field
2131      3017 6      LEAVE fld;
2132      3018 5      END;
2133      3019 5
2134      3020 5      ! Initialize flags
2135      3021 5      NAM_SET = FALSE;
2136      3022 5      POS_SET = FALSE;
2137      3023 5      SIZ_SET = FALSE;
2138      3024 5      DIG_SET = FALSE;
2139      3025 5      DATA_TYPE = 0;
2140      3026 5
2141      3027 5      ! While there are more field sub-specifications
2142      3028 5      WHILE TRUE DO
2143      3029 6      BEGIN
2144      3030 6
2145      3031 6      KYW_LINE = .SPC[ LINE];
2146      3032 6      SUB_KYW_TYPE = GET_SUB_SPEC( SPC[0], SUB_SPC[0] );
2147      3033 6
2148      3034 6      ! Check the sub-specification
2149      3035 6      !
2150      3036 6      SELECTONE .SUB_KYW_TYPE OF
2151      3037 6      SET
2152      3038 6
2153      3039 6      [KW_NAME]: ! Field name
2154      3040 7      BEGIN
2155      3041 7
2156      3042 7      ! Name should not already be defined and
2157      3043 7      ! Length must be greater than zero
2158      3044 7      IF .SUB_SPC[ LEN] LEQ 0 OR
2159      3045 7      .NAM_SET
2160      3046 7      THEN
2161      3047 8      BEGIN
2162      3048 8      ERR ( .KYW_LINE, SOR$_SPCIVF );      ! Invalid field
2163      3049 8      LEAVE fld;
2164      3050 7      END;
2165      3051 7
2166      3052 7      ! Look for the name in the field symbol table
2167      3053 7      IF SEARCH_TABLE( SUB_SPC[0], INDEX,
2168      3054 7      FLD_SYM[0, BASE_], .CAL[CA_FDT_SIZ]-1 ) EQL SUCCESS
2169      3055 7      THEN
2170      3056 7      ! Name has already been used
2171      3057 8      BEGIN
2172      3058 8      ERR ( .SUB_SPC[ LINE], SOR$_SPCIVF );      ! Invalid field
2173      3059 8      LEAVE fld;
2174      3060 7      END;
2175      3061 7
2176      3062 7      ! Put the length/address in the field symbol table entry
2177      3063 7      ! Put the fdt index in the field symbol table
2178      3064 8      BEGIN
2179      3065 8      LOCAL FLD_SYM_LOC: REF SYM_TAB [1];
2180      3066 8      FLD_SYM_LOC = FLD_SYM[ .CAL[CA_FDT_SIZ]-1, BASE_];
2181      3067 8      FLD_SYM_LOC[0, SYM_NAM_LEN] = .SUB_SPC[ LEN];
2182      3068 8      FLD_SYM_LOC[0, SYM_NAM_ADR] = .SUB_SPC[ PTR];
2183      3069 8      FLD_SYM_LOC[0, SYM_INDEX] = .CAL[CA_FDT_SIZ]-1;
2184      3070 7      END;
2185      3071 7
2186      3072 7      ! Name has been set
```


2187 3073 7
2188 3074 7
2189 3075 7
2190 3076 7
2191 3077 7
2192 3078 6
2193 3079 6
2194 3080 6
2195 3081 7
2196 3082 7
2197 3083 7
2198 3084 7
2199 3085 7
2200 3086 7
2201 3087 7
2202 3088 7
2203 3089 7
2204 3090 7
2205 3091 7
2206 3092 8
2207 3093 8
2208 3094 8
2209 3095 7
2210 3096 7
2211 3097 7
2212 3098 7
2213 3099 7
2214 3100 7
2215 3101 7
2216 3102 7
2217 3103 7
2218 3104 7
2219 3105 7
2220 3106 7
2221 3107 6
2222 3108 6
2223 3109 6
2224 3110 7
2225 3111 7
2226 3112 7
2227 3113 7
2228 3114 7
2229 3115 7
2230 3116 7
2231 3117 7
2232 3118 7
2233 3119 7
2234 3120 7
2235 3121 7
2236 3122 7
2237 3123 8
2238 3124 8
2239 3125 8
2240 3126 7
2241 3127 7
2242 3128 7
2243 3129 7

```
NAM_SET = TRUE;

! Nothing is left
SUB_SPC[LEN] = 0;

END;

[KW_SIZE, KW_DIGI]:
BEGIN
LOCAL
STATUS;                                ! Return status

! Get the number
STATUS = CVT_ATB( 10, SUB_SPC[0], NUMBER );

! Size should not be set and
! Positive numbers only
IF .STATUS EQL FAIL OR .SIZ_SET
THEN
BEGIN
ERR ( .KYW_LINE, SOR$_SPCIVF );        ! Invalid field
LEAVE fld;
END;

! Size has been set
SIZ_SET = TRUE;

! Place value in size field of FDT
FDT[0, FDT_FLD_SIZ] = .NUMBER;
%IF %BLISS(BLISS32) %THEN
FDT[0, FDT_SCALE] = 0;
%FI

DIG_SET = (.SUB_KYW_TYPE EQL KW_DIGI);
END;

[KW_POSI]:
BEGIN
LOCAL
STATUS;                                ! Return status

! Get the number
STATUS = CVT_ATB( 10, SUB_SPC[0], NUMBER );

! Position should not already be set and
! Number must be greater than zero
IF .STATUS EQL FAIL OR
.POS SET OR
.NUMBER LEQ 0
THEN
BEGIN
ERR ( .KYW_LINE, SOR$_SPCIVF );        ! Invalid field
LEAVE fld;
END;

! Position has been set
POS_SET = TRUE;
```


2244	3130	7
2245	3131	7
2246	3132	7
2247	3133	7
2248	3134	7
2249	3135	6
2250	3136	6
2251	3137	6
2252	3138	7
2253	3139	7
2254	3140	7
2255	3141	6
2256	3142	6
2257	3143	6
2258	3144	7
2259	3145	7
2260	3146	7
2261	3147	6
2262	3148	6
2263	3149	6
2264	3150	6
2265	3151	6
2266	3152	6
2267	3153	7
2268	3154	7
2269	3155	7
2270	3156	6
2271	3157	6
2272	3158	6
2273	3159	6
2274	3160	6
2275	3161	6
2276	3162	7
2277	3163	7
2278	3164	7
2279	3165	6
2280	3166	6
2281	3167	6
2282	3168	7
2283	3169	7
2284	3170	7
2285	3171	6
2286	3172	6
2287	3173	6
2288	3174	7
2289	3175	7
2290	3176	7
2291	3177	6
2292	3178	6
2293	3179	6
2294	3180	7
2295	3181	7
2296	3182	7
2297	3183	6
2298	3184	6
2299	3185	6
2300	3186	7

```
! Place value in position field of FDT
! Make it zero based
FDT[0, FDT_FLD_POS] = .NUMBER - 1;

END;

[KW_ASCF]:
BEGIN
  TYP = DT_AF;
  DATA_TYPE = .DATA_TYPE OR OTHERS_BIT;
END;

[KW_ASCZ]:
BEGIN
  TYP = DT_AZ;
  DATA_TYPE = .DATA_TYPE OR OTHERS_BIT;
END;

[KW_BINA]:
  DATA_TYPE = .DATA_TYPE OR BINARY_BIT;

[KW_CHAR]:
BEGIN
  TYP = DT_T;
  DATA_TYPE = .DATA_TYPE OR OTHERS_BIT;
END;

[KW_DEC1]:
  DATA_TYPE = .DATA_TYPE OR DECIMA_BIT;

[KW_DIBO]:
BEGIN
  TYP = DT_DB;
  DATA_TYPE = .DATA_TYPE OR OTHERS_BIT;
END;

[KW_FFLT]:
BEGIN
  TYP = DT_F;
  DATA_TYPE = .DATA_TYPE OR OTHERS_BIT;
END;

[KW_DFLT]:
BEGIN
  TYP = DT_D;
  DATA_TYPE = .DATA_TYPE OR OTHERS_BIT;
END;

[KW_GFLT]:
BEGIN
  TYP = DT_G;
  DATA_TYPE = .DATA_TYPE OR OTHERS_BIT;
END;

[KW_HFLT]:
BEGIN
```



```

: 2301      3187  7      TYP = DT_H;
: 2302      3188  7      DATA_TYPE = .DATA_TYPE OR OTHERS_BIT;
: 2303      3189  6      END;
: 2304      3190  6
: 2305      3191  6      [KW_PACK]:
: 2306      3192  7      BEGIN
: 2307      3193  7      TYP = DT_P;
: 2308      3194  7      DATA_TYPE = .DATA_TYPE OR OTHERS_BIT;
: 2309      3195  6      END;
: 2310      3196  6
: 2311      3197  6      [KW_ZONE]:
: 2312      3198  7      BEGIN
: 2313      3199  7      TYP = DT_NZ;
: 2314      3200  7      DATA_TYPE = .DATA_TYPE OR OTHERS_BIT;
: 2315      3201  6      END;
: 2316      3202  6
: 2317      3203  6      [KW_SIGN]:
: 2318      3204  6      DATA_TYPE = .DATA_TYPE OR SIGNED_BIT;
: 2319      3205  6
: 2320      3206  6      [KW_UNSI]:
: 2321      3207  6      DATA_TYPE = .DATA_TYPE OR UNSIGN_BIT;
: 2322      3208  6
: 2323      3209  6      [KW_LEAD]:
: 2324      3210  6      DATA_TYPE = .DATA_TYPE OR LEADIN_BIT;
: 2325      3211  6
: 2326      3212  6      [KW_TRAI]:
: 2327      3213  6      DATA_TYPE = .DATA_TYPE OR TRAILI_BIT;
: 2328      3214  6
: 2329      3215  6      [KW_OVER]:
: 2330      3216  6      DATA_TYPE = .DATA_TYPE OR OVERPU_BIT;
: 2331      3217  6
: 2332      3218  6      [KW_SEPA]:
: 2333      3219  6      DATA_TYPE = .DATA_TYPE OR SEPARA_BIT;
: 2334      3220  6
: 2335      3221  6      [OTHERWISE]:          ! Bad sub-specification
: 2336      3222  7      BEGIN
: 2337      3223  7      ERR ( .KYW_LINE, SOR$_SPCIVF );
: 2338      3224  7      LEAVE fld;
: 2339      3225  6      END;
: 2340      3226  6
: 2341      3227  6      TES;
: 2342      3228  6
: 2343      3229  6      ! Check for extraneous matter
: 2344      3230  6      IF .SUB_SPC[_LEN] GTR 0
: 2345      3231  6      THEN
: 2346      3232  6      ERR_( .SUB_SPC[_LINE], SOR$_SPCIVF );          ! Invalid field
: 2347      3233  6
: 2348      3234  6      IF SKIP_COMMA( SPC[0] ) EQL FAIL THEN EXITLOOP;
: 2349      3235  5      END;
: 2350      3236  5
: 2351      3237  5      ! Determine final data type, which bits were set above?
: 2352      3238  5
: 2353      3239  5      IF .DATA_TYPE NEQ 0
: 2354      3240  5      THEN
: 2355      3241  6      BEGIN
: 2356      3242  6      MACRO
: 2357      M 3243  6      DEF_(X,Y) =          ! Macro to apply defaults
```



```
2358  
2359  
2360  
2361  
2362  
2363  
2364  
2365  
2366  
2367  
2368  
2369  
2370  
2371  
2372  
2373  
2374  
2375  
2376  
2377  
2378  
2379  
2380  
2381  
2382  
2383  
2384  
2385  
2386  
2387  
2388  
2389  
2390  
2391  
2392  
2393  
2394  
2395  
2396  
2397  
2398  
2399  
2400  
2401  
2402  
2403  
2404  
2405  
2406  
2407  
2408  
2409  
2410  
2411  
2412  
2413  
2414
```

```
MEM 3244 6  
3245 6  
3246 6  
3247 6  
3248 6  
3249 6  
3250 6  
3251 6  
3252 6  
3253 6  
3254 7  
3255 7  
3256 7  
3257 6  
3258 5  
3259 5  
3260 5  
3261 5  
3262 5  
3263 5  
3264 5  
3265 5  
3266 5  
3267 5  
3268 5  
3269 5  
3270 5  
3271 5  
3272 5  
3273 5  
3274 5  
3275 5  
3276 5  
3277 5  
3278 5  
3279 5  
3280 5  
3281 5  
3282 5  
3283 5  
3284 5  
3285 5  
3286 5  
3287 5  
3288 5  
3289 6  
3290 5  
3291 5  
3292 5  
3293 5  
3294 5  
3295 5  
3296 5  
3297 5  
3298 6  
3299 6  
3300 6
```

```
BEGIN  
IF (.DATA_TYPE AND (X)) EQL 0  
THEN  
    DATA_TYPE = .DATA_TYPE OR (Y);  
END %;  
IF (.DATA_TYPE AND OTHERS_BIT) EQL 0  
THEN  
    DEF (SIGNED_BIT OR UNSIGN_BIT, SIGNED_BIT);  
IF (.DATA_TYPE AND DECIMA_BIT) NEQ 0  
THEN  
    BEGIN  
        DEF (UNSIGN_BIT OR TRAILI_BIT OR LEADIN_BIT, TRAILI_BIT);  
        DEF (UNSIGN_BIT OR SEPARA_BIT OR OVERPU_BIT, OVERPU_BIT);  
    END;  
END;  
  
! BINARY_BIT = %X'0001', ! Binary  
! DECIMA_BIT = %X'0002', ! Decimal  
! SIGNED_BIT = %X'0004', ! Signed  
! UNSIGN_BIT = %X'0008', ! Unsigned  
! TRAILI_BIT = %X'0010', ! Trailing sign  
! LEADIN_BIT = %X'0020', ! Leading sign  
! SEPARA_BIT = %X'0040', ! Separate sign  
! OVERPU_BIT = %X'0080', ! Overpunched sign  
! OTHERS_BIT = %X'0100', ! Other explicit data types  
  
SELECTONE .DATA_TYPE OF  
SET  
[ %X'0000' ]: TYP = DT_T; ! Character  
[ %X'0100' ]: 0; ! Do nothing, already set above  
[ %X'0005' ]: TYP = DT_B; ! Signed binary  
[ %X'0009' ]: TYP = DT_U; ! Unsigned binary  
[ %X'000A' ]: TYP = DT_NU; ! Unsigned decimal  
[ %X'0096' ]: TYP = DT_NRO; ! Decimal trailing overpunch  
[ %X'0056' ]: TYP = DT_NR; ! Decimal trailing separate  
[ %X'00A6' ]: TYP = DT_NLO; ! Decimal leading overpunch  
[ %X'0066' ]: TYP = DT_NL; ! Decimal leading separate  
[ OTHERWISE ]: ! Bad data type  
    ERR_( .KYW_LINE, SOR$_SPCIVD );  
TES;  
  
FDT[0, FDT_TYPE] = .TYP;  
IF .TYP LSS 0 THEN ERR_( .KYW_LINE, SOR$_SPCIVF );  
  
! Check for missing stuff  
IF NOT (.NAM_SET AND .POS_SET AND .SIZ_SET)  
THEN  
    ERR_( .KYW_LINE, SOR$_SPCIVF ); ! invalid field  
  
! If the key is decimal, zoned or packed, verify that the length  
! was specified in digits (on VAX).  
! Convert DIGITS to bytes, except for VAX packed.  
! If the key is not decimal or packed, digits can't be used.  
BEGIN  
MACRO  
    SIZE = FDT[0, FDT_FLD_SIZ] %;
```



```
IF (.DATA TYPE AND DECIMA_BIT) NEQ 0
  OR .TYP EQL DT_NZ
  OR .TYP EQL DT_DB
THEN
  BEGIN
    IF NOT .DIG_SET
    THEN
      %IF %BLISS(BLISS32)
      %THEN ERR_(.KYW_LINE, SORS$_SPCIVF )
      %ELSE 0
      %FI
    ELSE
      IF (.DATA_TYPE AND SEPARA_BIT) NEQ 0 THEN SIZE = .SIZE + 1;
    END
  ELSE IF .TYP EQL DT_P
  THEN
    BEGIN
      %IF %BLISS(BLISS32)
      %THEN IF NOT .DIG_SET THEN ERR_(.KYW_LINE, SORS$_SPCIVF );
      %FI
      %IF %BLISS(BLISS16)
      %THEN SIZE = (.SIZE + 1) / 2;
      %FI
    END
  ELSE
    IF .DIG_SET THEN ERR_(.KYW_LINE, SORS$_SPCIVF );
  END;
END;

%IF %BLISS(BLISS32) %THEN
[KW_CDD]:
BEGIN
  EXTERNAL ROUTINE
  CRE_EXTRACT_RECORD: CA_LINKAGE;
  LOCAL
  STATUS;
  IF .SPC[LEN] GEQ 2 THEN
    IF CH$RCHAR(.SPC[_PTR]) EQL C_QUOTE AND
    CH$RCHAR(.SPC[_PTR]+.SPC[_LEN]-1) EQL C_QUOTE
  THEN
    BEGIN
      SPC[_PTR] = .SPC[_PTR]+1;
      SPC[_LEN] = .SPC[_LEN]-2;
    END;
    STATUS = CRE_EXTRACT_RECORD( SPC[0], FLD_SYM[0,BASE_], .KYW_LINE );
    IF ERROR_(.STATUS )
    THEN
      ERR_(.KYW_LINE, .STATUS );
    END;
  %FI
[OTHERWISE]:
  0;
TES;
! Move pointer to next specification
```


P2\$PAS
T03-038

N 3
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 75
(21)

: 2472 3358 4
: 2473 3359 4
: 2474 3360 3
: 2475 3361 3
: 2476 3362 2
: 2477 3363 2
: 2478 3364 2
: 2479 3365 1

IF SKIP_IGNORED(BUFF[0]) EQL FAIL THEN EXITLOOP;

END;

END;

! Pass 2

RETURN SUCCESS;

END;

.TITLE P2\$PAS
.IDENT \T03-038\

.EXTRN EC\$ERR_CNTL, ST\$SEARCH_TABLE
.EXTRN CAS\$CVT_ATB, SAS\$PC_ALLOC
.EXTRN SI\$SKIP_IGNORED
.EXTRN SC\$SKIP_COMMA, GNS\$GET_NEXT_SPEC
.EXTRN GSS\$GET_SUB_SPEC
.EXTRN CRE_EXTRACT_RECORD

.PSECT SOR\$RO_CODE, NOWRT, SHR, PIC, 2

.ENTRY P2\$PASS_TWO, Save R2,R3,R4,R5,R6,R7,R8,R9,- : 2902
R10
SUBL2 #48, SP
MOVAB 272(CA), R2 : 2940
TSTL (R2)
BNEQ 1\$
MOVL 296(CA), (R2) : 2943
CLRB 255(CA) : 2944
BRB 2\$: 2940
MOVL (R2), FROM_PTR : 2951
MOVZBL 255(CA), R0 : 2952
MULL3 #6, R0, -(SP)
CALLS #1, SAS\$PC_ALLOC
MOVL R0, (R2)
MOVZBL 255(CA), R0 : 2953
MULL2 #6, R0
MOVC3 R0, (FROM_PTR), @272(CA)
MOVL START, R0 : 2957
MOVQ (R0), BUFF
MOVL 8(R0), BUFF+8 : 2959
PUSHL #6 : 2964
CALLS #1, SAS\$PC_ALLOC
MOVL R0, FDT
INCB 255(CA) : 2965
MOVL #14, (FDT) : 2966
MNEGW #1, 4(FDT) : 2969
MOVL BUFF+8, KYW_LINE : 2978
PUSHAB SPC : 2979
PUSHAB BUFF
CALLS #2, GNS\$GET_NEXT_SPEC
MOVL R0, KYW_TYPE
CMLP KYW_TYPE, #19 : 2985
BEQL 4\$
BRW 62\$
PUSHL #6 : 3009

07FC 00000
5E 30 C2 00002
52 0110 CB 9E 00005
62 0128 62 D5 0000A
00FF 0B 12 0000C
CB D0 0000E
24 11 00013
53 62 D0 00017 1\$:
50 00FF CB 9A 00019
7E 00000000G 50 06 C5 00021
00 01 FB 00025
62 50 D0 0002C
50 00FF CB 9A 0002F
50 06 C4 00034
0110 DB 63 50 28 00037 2\$:
50 AC D0 0003D
24 AE 60 7D 00041
2C AE 08 A0 D0 00045
00000000G 00 06 DD 0004A
53 01 FB 0004C
00FF 50 D0 00053
63 0E D0 0005A
04 A3 01 AE 0005D
56 2C AE D0 00061 3\$:
18 AE 9F 00065
28 AE 9F 00068
00000000G 00 02 FB 0006B
6E 50 D0 00072
13 6E D1 00075
03 13 00078
02E1 31 0007A
06 DD 0007D 4\$:

00000000G	00	01	FB	0007F	CALLS	#1, SASSPC_ALLOC	:		
	53	50	D0	00086	MOVL	R0, FDT	:		
		CB	96	00089	INCB	255(CA)	:	3010	
FF	8F	00FF	CB	91	0008D	CMPB	255(CA), #255	3013	
		29	1E	00093	BGEQU	6\$	:		
		58	7C	00095	CLRQ	POS_SET	:	3022	
		57	D4	00097	CLRL	SIZ_SET	:	3023	
		5A	D4	00099	CLRL	DIG_SET	:	3024	
		52	D4	0009B	CLRL	DATA_TYPE	:	3025	
	56	20	AE	D0	0009D	5\$: MOVL	SPC+8, KYW_LINE	3031	
		0C	AE	9F	000A1	PUSHAB	SUB_SPC	3032	
		1C	AE	9F	000A4	PUSHAB	SPC		
00000000G	00	02	FB	000A7	CALLS	#2, GSS\$GET_SUB_SPEC	:		
	55	50	D0	000AE	MOVL	R0, SUB_KYW_TYPE	:		
	14	55	D1	000B1	CMPL	SUB_KYW_TYPE, #20	:	3039	
		5E	12	000B4	BNEQ	9\$	:		
		0C	AE	D5	000B6	TSTL	SUB_SPC	3044	
		03	15	000B9	BLEQ	6\$	:		
	09	59	E9	000BB	BLBC	NAM_SET, 7\$	:	3045	
		001C8194	8F	DD	000BE	6\$: PUSHL	#1868180	3048	
		02CF	31	000C4	BRW	64\$	:		
	7E	00FF	CB	9A	000C7	7\$: MOVZBL	255(CA), -(SP)	3054	
			6E	D7	000CC	DECL	(SP)		
		08	AC	DD	000CE	PUSHL	FLD_SYM		
		0C	AE	9F	000D1	PUSHAB	INDEX	3053	
		18	AE	9F	000D4	PUSHAB	SUB_SPC		
00000000G	00	04	FB	000D7	CALLS	#4, ST\$SEARCH_TABLE	:	3054	
	01	50	D1	000DE	CMPL	R0, #1	:		
		0C	12	000E1	BNEQ	8\$	:		
		001C8194	8F	DD	000E3	PUSHL	#1868180	3058	
		18	AE	DD	000E9	PUSHL	SUB_SPC+8		
		02A9	31	000EC	BRW	65\$	:		
	50	00FF	CB	9A	000EF	8\$: MOVZBL	255(CA), R0	3066	
	50		06	C4	000F4	MULL2	#6, R0		
	50	08	AC	C0	000F7	ADDL2	FLD_SYM, R0		
			50	D7	000FB	DECL	FLD_SYM_LOC		
	70	0C	AE	90	000FD	MOVW	SUB_SPC, -(FLD_SYM_LOC)	3067	
	70	10	AE	D0	00101	MOVL	SUB_SPC+4, -(FLD_SYM_LOC)	3068	
05	A0	00FF	CB	01	83	00105	SUBB3	#1, 255(CA), 5(FLD_SYM_LOC)	3069
			01	D0	0010C	MOVL	#1, NAM_SET	3073	
		0C	AE	D4	0010F	CLRL	SUB_SPC	3076	
			7E	11	00112	BRB	18\$	3036	
	15		55	D1	00114	9\$: CMPL	SUB_KYW_TYPE, #21	3080	
			05	13	00117	BEQL	10\$		
	17		55	D1	00119	CMPL	SUB_KYW_TYPE, #23		
			2F	12	0011C	BNEQ	12\$		
		08	AE	9F	0011E	10\$: PUSHAB	NUMBER	3086	
		10	AE	9F	00121	PUSHAB	SUB_SPC		
			0A	DD	00124	PUSHL	#10		
00000000G	00	03	FB	00126	CALLS	#3, CASCVT_ATB	:	3090	
		50	D5	0012D	TSTL	STATUS	:		
		8D	13	0012F	BEQL	6\$	:		
	8A	57	E8	00131	BLBS	SIZ_SET, 6\$	:		
			01	D0	00134	MOVL	#1, SIZ_SET	3098	
	04	A3	08	AE	B0	00137	MOVW	NUMBER, -4(FDT)	3101
			01	A3	94	0013C	CLRB	1(FDT)	3103
			50	D4	0013F	CLRL	R0	3106	

17	55	D1	00141	CMPL	SUB_KYW_TYPE, #23	:	
	02	12	00144	BNEQ	11\$	:	
	50	D6	00146	INCL	R0	:	
5A	50	D0	00148	11\$: MOVL	R0, DIG_SET	:	
	59	11	0014B	BRB	21\$	:	3036
16	55	D1	0014D	12\$: CMPL	SUB_KYW_TYPE, #22	:	3109
	29	12	00150	BNEQ	15\$	:	
08	AE	9F	00152	PUSHAB	NUMBER	:	3115
10	AE	9F	00155	PUSHAB	SUB_SPC	:	
	0A	DD	00158	PUSHL	#10	:	
00000000G 00	03	FB	0015A	CALLS	#3, CASCVT_ATB	:	
	50	D5	00161	TSTL	STATUS	:	3119
	03	13	00163	BEQL	13\$	:	
03	58	E9	00165	BLBC	POS_SET, 14\$	:	3120
	FF	31	00168	13\$: BRW	6\$	:	
08	AE	D5	0016B	14\$: TSTL	NUMBER	:	3121
	F8	15	0016E	BLEQ	13\$	:	
02	01	D0	00170	MOVL	#1, POS_SET	:	3129
A3	01	A3	00173	SUBW3	#1, NUMBER, 2(FDT)	:	3133
08	7B	11	00179	BRB	30\$	:	3036
	55	D1	0017B	15\$: CMPL	SUB_KYW_TYPE, #24	:	3137
	05	13	0017E	BEQL	16\$	:	
19	55	D1	00180	CMPL	SUB_KYW_TYPE, #25	:	3143
	05	12	00183	BNEQ	17\$	:	
54	01	8E	00185	16\$: MNEGB	#1, TYP	:	3145
	5D	11	00188	BRB	28\$	:	3146
1A	55	D1	0018A	17\$: CMPL	SUB_KYW_TYPE, #26	:	3149
	05	12	0018D	BNEQ	19\$	:	
52	01	88	0018F	BISB2	#1, DATA_TYPE	:	3150
	76	11	00192	18\$: BRB	33\$	:	
1B	55	D1	00194	19\$: CMPL	SUB_KYW_TYPE, #27	:	3152
	05	12	00197	BNEQ	20\$	:	
54	0E	90	00199	MOVB	#14, TYP	:	3154
	49	11	0019C	BRB	28\$	:	3155
1C	55	D1	0019E	20\$: CMPL	SUB_KYW_TYPE, #28	:	3158
	05	12	001A1	BNEQ	22\$	:	
52	02	88	001A3	BISB2	#2, DATA_TYPE	:	3159
	77	11	001A6	21\$: BRB	36\$	:	
1D	55	D1	001A8	22\$: CMPL	SUB_KYW_TYPE, #29	:	3161
	D8	13	001AB	BEQL	16\$	:	
1E	55	D1	001AD	CMPL	SUB_KYW_TYPE, #30	:	3167
	05	12	001B0	BNEQ	23\$	:	
54	0A	90	001B2	MOVB	#10, TYP	:	3169
	30	11	001B5	BRB	28\$	:	3170
1F	55	D1	001B7	23\$: CMPL	SUB_KYW_TYPE, #31	:	3173
	05	12	001BA	BNEQ	24\$	:	
54	0B	90	001BC	MOVB	#11, TYP	:	3175
	26	11	001BF	BRB	28\$	:	3176
20	55	D1	001C1	24\$: CMPL	SUB_KYW_TYPE, #32	:	3179
	05	12	001C4	BNEQ	25\$	:	
54	1B	90	001C6	MOVB	#27, TYP	:	3181
	1C	11	001C9	BRB	28\$	:	3182
21	55	D1	001CB	25\$: CMPL	SUB_KYW_TYPE, #33	:	3185
	05	12	001CE	BNEQ	26\$	:	
54	1C	90	001D0	MOVB	#28, TYP	:	3187
	12	11	001D3	BRB	28\$	:	3188
22	55	D1	001D5	26\$: CMPL	SUB_KYW_TYPE, #34	:	3191

54	05	12	001D8	BNEQ	27\$				
	15	90	001DA	MOVB	#21, TYP				3193
	08	11	001DD	BRB	28\$				3194
23	55	D1	001DF	27\$:	CMPL	SUB_KYW_TYPE, #35			3197
	0A	12	001E2	BNEQ	29\$				
54	14	90	001E4	MOVB	#20, TYP				3199
52	8F	A8	001E7	28\$:	BISW2	#256, DATA_TYPE			3200
	3F	11	001EC	BRB	39\$				3036
24	55	D1	001EE	29\$:	CMPL	SUB_KYW_TYPE, #36			3203
	05	12	001F1	BNEQ	31\$				
52	04	88	001F3	BISB2	#4, DATA_TYPE				3204
	35	11	001F6	30\$:	BRB	39\$			
25	55	D1	001F8	31\$:	CMPL	SUB_KYW_TYPE, #37			3206
	05	12	001FB	BNEQ	32\$				
52	08	88	001FD	BISB2	#8, DATA_TYPE				3207
	2B	11	00200	BRB	39\$				
26	55	D1	00202	32\$:	CMPL	SUB_KYW_TYPE, #38			3209
	05	12	00205	BNEQ	34\$				
52	20	88	00207	BISB2	#32, DATA_TYPE				3210
	21	11	0020A	33\$:	BRB	39\$			
27	55	D1	0020C	34\$:	CMPL	SUB_KYW_TYPE, #39			3212
	05	12	0020F	BNEQ	35\$				
52	10	88	00211	BISB2	#16, DATA_TYPE				3213
	17	11	00214	BRB	39\$				
28	55	D1	00216	35\$:	CMPL	SUB_KYW_TYPE, #40			3215
	06	12	00219	BNEQ	37\$				
52	80	8F	0021B	BISB2	#128, DATA_TYPE				3216
	0C	11	0021F	36\$:	BRB	39\$			
29	55	D1	00221	37\$:	CMPL	SUB_KYW_TYPE, #41			3218
	03	13	00224	BEQL	38\$				
	FE	95	31	00226	BRW	6\$			
52	40	8F	88	00229	38\$:	BISB2	#64, DATA_TYPE		3219
	0C	AE	D5	0022D	39\$:	TSTL	SUB_SPC		3230
		18	15	00230	BLEQ	40\$			
	001C8194	8F	DD	00232	PUSHL	#1868180			3232
		18	AE	DD	00238	PUSHL	SUB_SPC+8		
00000000G	00	02	FB	0023B	CALLS	#2, EC\$ERR_CNTL			
	01	50	D1	00242	CMPL	R0, #1			
		03	13	00245	BEQL	40\$			
		016F	31	00247	BRW	69\$			
		18	AE	9F	0024A	40\$:	PUSHAB	SPC	3234
00000000G	00	01	FB	0024D	CALLS	#1, SC\$SKIP_COMMA			
		50	D5	00254	TSTL	R0			
		03	13	00256	BEQL	41\$			
		FE	42	31	00258	BRW	5\$		
		52	D5	0025B	41\$:	TSTL	DATA_TYPE		3239
		22	13	0025D	BEQL	44\$			
08	52	08	E0	0025F	BBS	#8, DATA_TYPE	42\$		3249
	0C	52	93	00263	BITB	DATA_TYPE, #12			3251
		03	12	00266	BNEQ	42\$			
	52	04	88	00268	BISB2	#4, DATA_TYPE			
12	52	01	E1	0026B	42\$:	BBC	#1, DATA_TYPE	44\$	3252
	38	52	93	0026F	BITB	DATA_TYPE, #56			3255
		03	12	00272	BNEQ	43\$			
		10	88	00274	BISB2	#16, DATA_TYPE			
	c8	52	93	00277	43\$:	BITB	DATA_TYPE, #200		3256
		8F	04	12	0027B	BNEQ	44\$		

	52	80	8F	88	0027D		BISB2	#128, DATA_TYPE		
			52	D5	00281	44\$:	TSTL	DATA_TYPE		3272
			05	12	00283		BNEQ	45\$		
	54		0E	90	00285		MOVB	#14, TYP		
			73	11	00288		BRB	53\$		
00000100	8F		52	D1	0028A	45\$:	CMPL	DATA_TYPE, #256		3273
			6A	13	00291		BEQL	53\$		
	05		52	D1	00293		CMPL	DATA_TYPE, #5		3274
			05	12	00296		BNEQ	46\$		
	54		06	90	00298		MOVB	#6, TYP		
			60	11	0029B		BRB	53\$		
	09		52	D1	0029D	46\$:	CMPL	DATA_TYPE, #9		3275
			05	12	002A0		BNEQ	47\$		
	54		02	90	002A2		MOVB	#2, TYP		
			56	11	002A5		BRB	53\$		
	0A		52	D1	002A7	47\$:	CMPL	DATA_TYPE, #10		3276
			05	12	002AA		BNEQ	48\$		
	54		0F	90	002AC		MOVB	#15, TYP		
			4C	11	002AF		BRB	53\$		
00000096	8F		52	D1	002B1	48\$:	CMPL	DATA_TYPE, #150		3277
			05	12	002B8		BNEQ	49\$		
	54		13	90	002BA		MOVB	#19, TYP		
			3E	11	002BD		BRB	53\$		
00000056	8F		52	D1	002BF	49\$:	CMPL	DATA_TYPE, #86		3278
			05	12	002C6		BNEQ	50\$		
	54		12	90	002C8		MOVB	#18, TYP		
			30	11	002CB		BRB	53\$		
000000A6	8F		52	D1	002CD	50\$:	CMPL	DATA_TYPE, #166		3279
			05	12	002D4		BNEQ	51\$		
	54		11	90	002D6		MOVB	#17, TYP		
			22	11	002D9		BRB	53\$		
00000066	8F		52	D1	002DB	51\$:	CMPL	DATA_TYPE, #102		3280
			05	12	002E2		BNEQ	52\$		
	54		10	90	002E4		MOVB	#16, TYP		
			14	11	002E7		BRB	53\$		
		001C819C	8F	DD	002E9	52\$:	PUSHL	#1868188		3282
			56	DD	002EF		PUSHL	KYW_LINE		
00000000G	00		02	FB	002F1		CALLS	#2, EC\$ERR_CNTL		
	01		50	D1	002F8		CMPL	R0, #1		
			34	12	002FB		BNEQ	56\$		
	63		54	90	002FD	53\$:	MOVB	TYP, (FDT)		3285
			14	18	00300		BGEQ	54\$		3286
		001C8194	8F	DD	00302		PUSHL	#1868180		
			56	DD	00308		PUSHL	KYW_LINE		
00000000G	00		02	FB	0030A		CALLS	#2, EC\$ERR_CNTL		
	01		50	D1	00311		CMPL	R0, #1		
			1B	12	00314		BNEQ	56\$		
	06		59	E9	00316	54\$:	BLBC	NAM_SET, 55\$		3289
	03		58	E9	00319		BLBC	POS_SET, 55\$		
	14		57	E8	0031C		BLBS	SIZ_SET, 57\$		
		001C8194	8F	DD	0031F	55\$:	PUSHL	#1868180		3291
			56	DD	00325		PUSHL	KYW_LINE		
00000000G	00		02	FB	00327		CALLS	#2, EC\$ERR_CNTL		
	01		50	D1	0032E		CMPL	R0, #1		
			6F	12	00331	56\$:	BNEQ	66\$		
0B	52		01	E0	00333	57\$:	BBS	#1, DATA_TYPE, 58\$		3301
	14		54	91	00337		CMPB	TYP, #20		3302

	FF	8F		06	13	0033A	BEQL	58\$	:	
				54	91	0033C	CMPB	TYP, #1	:	3303
		16		0C	12	00340	BNEQ	59\$	:	
5B		52		5A	E9	00342	BLBC	DIG_SET, 61\$	:	3306
				06	E1	00345	BBC	#6, DATA_TYPE, 67\$	:	3313
			04	A3	B6	00349	INCW	4(FDT)	:	
		15		56	11	0034C	BRB	67\$	:	3301
				54	91	0034E	CMPB	TYP, #21	:	3315
		4E		05	12	00351	BNEQ	60\$	:	
				5A	E8	00353	BLBS	DIG_SET, 67\$	:	3319
		49		03	11	00356	BRB	61\$	:	
				5A	E9	00358	BLBC	DIG_SET, 67\$	:	3326
		3E		FD60	31	0035B	BRW	6\$	:	
				6E	D1	0035E	CMPL	KYW_TYPE, #62	:	3331
		02		41	12	00361	BNEQ	67\$	:	
			18	AE	D1	00363	CMPL	SPC, #2	:	3337
		22		19	19	00367	BLSS	63\$	:	
			1C	BE	91	00369	CMPB	@SPC+4, #34	:	3338
				13	12	0036D	BNEQ	63\$	:	
50	1C	AE		18	AE	C1	ADDL3	SPC, SPC+4, R0	:	3339
		22		FF	A0	91	CMPB	-1(R0), #34	:	
				07	12	00379	BNEQ	63\$	:	
			1C	AE	D6	0037B	INCL	SPC+4	:	3342
	18	AE		02	C2	0037E	SUBL2	#2, SPC	:	3343
				56	DD	00382	PUSHL	KYW_LINE	:	3345
			08	AC	DD	00384	PUSHL	FLD_SYM	:	
			20	AE	9F	00387	PUSHAB	SPC	:	
00000000G	00			03	FB	0038A	CALLS	#3, CRE_EXTRACT_RECORD	:	
	10			50	E8	00391	BLBS	STATUS, 67\$	:	3346
				50	DD	00394	PUSHL	STATUS	:	3348
00000000G	00			56	DD	00396	PUSHL	KYW_LINE	:	
	01			02	FB	00398	CALLS	#2, EC\$ERR_CNTL	:	
				50	D1	0039F	CMPL	R0, #1	:	
				15	12	003A2	BNEQ	69\$	:	
00000000G	00		24	AE	9F	003A4	PUSHAB	BUFF	:	3358
				01	FB	003A7	CALLS	#1, SI\$SKIP_IGNORED	:	
				50	D5	003AE	TSTL	R0	:	
				03	13	003B0	BEQL	68\$	:	
			FCAC	31	003B2		BRW	3\$	:	
	50			01	D0	003B5	MOVL	#1, R0	:	3364
				04	003B8		RET		:	
				50	D4	003B9	CLRL	R0	:	3365
				04	003BB		RET		:	

; Routine Size: 956 bytes, Routine Base: SOR\$R0_CODE + 0000

: 2480	3366	1	
: 2481	3367	1	! ELUDOM
: 2482	3368	1	!
: 2483	3369	1	END
: 2484	3370	0	ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
SORSRO_CODE	956	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	105	25	34	00:00.1
-\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	32	26	12	00:00.1
-\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LISS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI
)

2485	L 3371	0	MODULE P3\$PAS	(IDENT = 'T03-038' %REQUIRE('SRC\$:DEFS0')) =
2486	3377	1	BEGIN	
2487	3378	1	REQUIRE 'SRC\$:DKS';	
2488	3676	1	EXTERNAL ROUTINE	
2489	3677	1	ERR_CNTL:	CA_LINKAGE, ! Error control routine
2490	3678	1	SPC_ALLOC:	CA_LINKAGE, ! Allocate storage from the work area
2491	3679	1	SKIP_IGNORED,	! Skip ignored text characters
2492	3680	1	SKIP_COMMA,	! Skip commas and ignored stuff
2493	3681	1	GET_NEXT_SPEC,	! Get pointer and length of next specification
2494	3682	1	GET_SUB_SPEC,	! Get pointer and length of sub-specification
2495	3683	1	SEARCH_TABLE,	
2496	3684	1	PARSE_TEST:	CA_LINKAGE; ! Parse a test clause


```
2498 3685 1 GLOBAL ROUTINE P3$PASS_THREE(
2499 3686 1     START:      REF VECTOR[3],
2500 3687 1     CON_SYM:   REF CON_SYM_TAB,
2501 3688 1     P_CON_CNT:  REF VECTOR[1],
2502 3689 1     FCD_SYM:    REF SYM_TAB,
2503 3690 1     CDX_SYM:   REF SYM_TAB,
2504 3691 1     P_CDX_CNT: REF VECTOR[1]
2505 3692 1 );      CA_LINKAGE =
2506 3693 2 BEGIN
2507 3694 2 -----
2508 3695 2 Pass 3:  parse condition specifications.
2509 3696 2         set up test definition table,
2510 3697 2         condition symbol table and the
2511 3698 2         constant symbol table.
2512 3699 2         Build the TDT table
2513 3700 2 -----
2514 3701 2 BIND
2515 3702 2     CON_CNT = P_CON_CNT[0],
2516 3703 2     CDX_CNT = P_CDX_CNT[0];
2517 3704 2
2518 3705 2 LOCAL
2519 3706 2     KYW_TYPE,           ! Type of keyword
2520 3707 2     SUB_KYW_TYPE,
2521 3708 2     KYW_LINE,         ! Line number of keyword
2522 3709 2     BUFF:      VECTOR[3],
2523 3710 2     SPC:      VECTOR[3], ! Next specification length/pointer/line
2524 3711 2     SUB_SPC:  VECTOR[3], ! Sub specification length/pointer/line
2525 3712 2     COL:      VECTOR[3]; ! Collating sequence specification
2526 3713 2
2527 3714 2 ! Get the impure area
2528 3715 2 CA_AREA_( CA );
2529 3716 2
2530 3717 2 -----
2531 3718 2 Pass 3:  parse condition specifications.
2532 3719 2         set up test definition table,
2533 3720 2         condition symbol table and the
2534 3721 2         constant symbol table.
2535 3722 2         Build the TDT table
2536 3723 2 -----
2537 3724 2 BEGIN                                ! Pass 3
2538 3725 2
2539 3726 2 output_('pass 3');
2540 3727 2
2541 3728 2 ! Anchor condition definition table at beginning of available work area
2542 3729 2 ! Set address of condition definition table
2543 3730 2 CA[CA_TDT_ADR] = ALLOC_(0);
2544 3731 2
2545 3732 2 ! Initialize for pass 3
2546 3733 2 BUFF[_LEN] = .START[_LEN];
2547 3734 2 BUFF[_PTR] = .START[_PTR];
2548 3735 2 BUFF[_LINE] = .START[_LINE];
2549 3736 2
2550 3737 2 ! Exit loop when there are no remaining specifications
2551 3738 2 WHILE TRUE DO
2552 3739 2     BEGIN
2553 3740 2
2554 3741 2
```



```
2555 3742 4 LABEL
2556 3743 4 condition;
2557 3744 4
2558 3745 4 KYW_LINE = .BUFF[ LINE];
2559 3746 4 KYW_TYPE = GET_NEXT_SPEC( BUFF[0], SPC[0] );
2560 3747 4
2561 3748 4 ! Process the keyword (only conditions)
2562 3749 4 !
2563 3750 4 IF .KYW_TYPE EQL KW_COND
2564 3751 4 THEN
2565 3752 5 condition: BEGIN ! Make a entry in the test definition table
2566 3753 5 LOCAL
2567 3754 5 INDEX, ! Returned index in symbol table
2568 3755 5 NAM_SET, ! Name qualifier defined
2569 3756 5 TST_SET; ! Test qualifier defined
2570 3757 5
2571 3758 5 ! Too many entries in the named condition table?
2572 3759 5 IF .CDX_CNT GEQ MAX_CONDX
2573 3760 5 THEN
2574 3761 6 BEGIN
2575 3762 6 ERR ( .KYW_LINE, SORS_SPCIVX ); ! Invalid condition
2576 3763 6 LEAVE condition;
2577 3764 5 END;
2578 3765 5
2579 3766 5 NAM_SET = FALSE;
2580 3767 5 TST_SET = FALSE;
2581 3768 5
2582 3769 5 ! While there are more condition sub-specifications
2583 3770 5 WHILE TRUE DO
2584 3771 6 BEGIN
2585 3772 6
2586 3773 6 KYW_LINE = .SPC[ LINE];
2587 3774 6 SUB_KYW_TYPE = GET_SUB_SPEC( SPC[0], SUB_SPC[0] );
2588 3775 6
2589 3776 6 ! Check the sub-specification
2590 3777 6 !
2591 3778 6 SELECTONE .SUB_KYW_TYPE OF
2592 3779 6 SET
2593 3780 6
2594 3781 6 [KW_NAME]:
2595 3782 7 BEGIN
2596 3783 7
2597 3784 7 ! Name should not already be defined and
2598 3785 7 ! Length must be greater than zero
2599 3786 7 IF .SUB_SPC[LEN] LEQ 0
2600 3787 7 OR
2601 3788 7 .NAM_SET
2602 3789 7 THEN
2603 3790 8 BEGIN
2604 3791 8 ERR ( .KYW_LINE, SORS_SPCIVX ); ! Invalid condition
2605 3792 8 LEAVE condition;
2606 3793 7 END;
2607 3794 7
2608 3795 7 ! Look for the name in the condition symbol table
2609 3796 7 IF SEARCH TABLE( SUB_SPC[0], INDEX,
2610 3797 7 CDX_SYM[0, BASE_], .CDX_CNT ) EQL SUCCESS
2611 3798 7 THEN
```


.. 2612 3799 7
.. 2613 3800 8
.. 2614 3801 8
.. 2615 3802 8
.. 2616 3803 7
.. 2617 3804 7
.. 2618 3805 7
.. 2619 3806 7
.. 2620 3807 7
.. 2621 3808 7
.. 2622 3809 7
.. 2623 3810 7
.. 2624 3811 7
.. 2625 3812 7
.. 2626 3813 7
.. 2627 3814 7
.. 2628 3815 7
.. 2629 3816 7
.. 2630 3817 7
.. 2631 3818 6
.. 2632 3819 6
.. 2633 3820 6
.. 2634 3821 6
.. 2635 3822 7
.. 2636 3823 7
.. 2637 3824 7
.. 2638 3825 7
.. 2639 3826 7
.. 2640 3827 7
.. 2641 3828 7
.. 2642 3829 7
.. 2643 3830 7
.. 2644 3831 7
.. 2645 3832 7
.. 2646 3833 7
.. 2647 3834 7
.. 2648 3835 8
.. 2649 3836 8
.. 2650 3837 8
.. 2651 3838 7
.. 2652 3839 7
.. 2653 3840 7
.. 2654 3841 7
.. 2655 3842 7
.. 2656 3843 6
.. 2657 3844 6
.. 2658 3845 6
.. 2659 3846 7
.. 2660 3847 7
.. 2661 3848 7
.. 2662 3849 6
.. 2663 3850 6
.. 2664 3851 6
.. 2665 3852 6
.. 2666 3853 6
.. 2667 3854 5
.. 2668 3855 5

```
! Name has already been used
BEGIN
ERR ( .SUB_SPC[ _LINE], SOR$_SPCIVX );      ! Invalid condition
LEAVE condition;
END;

! Put the length/address in the condition symbol table entry
CDX_SYM[.CDX_CNT, SYM_NAM_LEN] = .SUB_SPC[ _LEN];
CDX_SYM[.CDX_CNT, SYM_NAM_ADR] = .SUB_SPC[ _PTR];

! Put the tdt index in the condition symbol table
CDX_SYM[.CDX_CNT, SYM_INDEX] = .CA[CA_TDT_SIZ];

! Increment the count of named condition
CDX_CNT = .CDX_CNT + 1;

! Name has been set
NAM_SET = TRUE;

END;

[KW_TEST]:
BEGIN
LOCAL
STATUS;                                     ! Return status

! Parse the test clause
STATUS = PARSE_TEST( SUB_SPC[0],
CON_SYM[0, BASE_], CON_CNT,
FLD_SYM[0, BASE_] );

! Test should not already be defined and test clause should
! Be greater then zero in length
IF .TST_SET OR .STATUS EQL FAIL
THEN
BEGIN
ERR ( .SUB_SPC[ _LINE], SOR$_SPCIVX );      ! Invalid condition
LEAVE condition;
END;

! Test has been set
TST_SET = TRUE;

END;

[OTHERWISE]:                               ! Neither "test" nor "name"
BEGIN
ERR ( .KYW_LINE, SOR$_SPCIVX ); ! Invalid condition
LEAVE condition;
END;

TES;

IF SKIP_COMMA( SPC[0] ) EQL FAIL THEN EXITLOOP;
END;
```



```

2669      3856 5      ! Make sure mandatory stuff is set
2670      3857 6      IF NOT (.NAM_SET AND .TST_SET)
2671      3858 5      THEN
2672      3859 6          BEGIN
2673      3860 6              ERR ( .KYW_LINE, SORS_SPCIVX ); ! Invalid condition
2674      3861 6              LEAVE condition;
2675      3862 5          END;
2676      3863 5
2677      3864 4      END;
2678      3865 4
2679      3866 4
2680      3867 4      ! Find the next specification, ignore comments and blanks
2681      3868 4      IF SKIP_IGNORED( BUFF[0] ) EQL FAIL THEN EXITLOOP;
2682      3869 4
2683      3870 3      END;
2684      3871 3
2685      3872 2      END;
2686      3873 2
2687      3874 2      ! Pass 3
2688      3875 1      RETURN SUCCESS;
                END;

```

.TITLE P3\$PAS
.IDENT \T03-038\

```
.EXTRN  EC$ERR_CNTL,  SASSPC_ALLOC
.EXTRN  SI$SKIP_IGNORED
.EXTRN  SC$SKIP_COMMA,  GNSS$GET_NEXT_SPEC
.EXTRN  GS$GET_SUB_SPEC
.EXTRN  ST$SEARCH_TABLE
.EXTRN  PT$PARSE_TEST
```

.PSECT SOR\$RO_CODE,NOWRT, SHR, PIC,2

```
.ENTRY P3$PASS_THREE, Save R2,R3,R4,R5,R6,R7
```

Address	Hex	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
---------	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

14	55	D1	0005C	CMPL	SUB_KYW_TYPE, #20	3781			
	4A	12	0005F	BNEQ	6\$				
	10	AE	D5	00061	TSTL	SUB_SPC	3786		
	03	15	00064	BLEQ	4\$				
0A	54	E9	00066	BLBC	NAM_SET, 5\$	3788			
001C81A4	8F	DD	00069	4\$: PUSHL	#1868196	3791			
	52	DD	0006F	PUSHL	KYW_LINE				
	65	11	00071	BRB	9\$				
	18	BC	DD	00073	5\$: PUSHL	@P CDX CNT	3797		
	14	AC	DD	00076	PUSHL	CDX_SYM			
	08	AE	9F	00079	PUSHAB	INDEX	3796		
	1C	AE	9F	0007C	PUSHAB	SUB_SPC			
00000000G	00	04	FB	0007F	CALLS	#4, ST\$SEARCH_TABLE	3797		
	01	50	D1	00086	CMPL	R0, #1			
		3D	13	00089	BEQL	7\$			
50	18	BC	06	C5	0008B	MULL3	#6, @P CDX CNT, R0	3806	
		50	14	AC	C0	00090	ADDL2	CDX_SYM, R0	
	04	A0	10	AE	90	00094	MOVB	SUB_SPC, 4(R0)	
	60	14	AE	D0	00099	MOVL	SUB_SPC+4, (R0)	3807	
05	A0	0100	CB	90	0009D	MOVB	256(CA), 5(R0)	3810	
		18	BC	D6	000A3	INCL	@P CDX CNT	3813	
	54		01	D0	000A6	MOVL	#1, NAM_SET	3816	
			37	11	000A9	BRB	10\$	3778	
	34		55	D1	000AB	6\$: CMPL	SUB_KYW_TYPE, #52	3820	
			49	12	000AE	BNEQ	12\$		
	7E	0C	AC	7D	000B0	MOVQ	P CON CNT, -(SP)	3829	
		08	AC	DD	000B4	PUSHL	CON_SYM		
		1C	AE	9F	000B7	PUSHAB	SUB_SPC	3827	
00000000G	00	04	FB	000BA	CALLS	#4, PT\$PARSE_TEST	3829		
			53	E8	000C1	BLBS	TST_SET, 7\$	3833	
			50	D5	000C4	TSTL	STATUS		
			0B	12	000C6	BNEQ	8\$		
	001C81A4	8F	DD	000C8	7\$: PUSHL	#1868196	3836		
		1C	AE	DD	000CE	PUSHL	SUB_SPC+8		
			05	11	000D1	BRB	9\$		
	53		01	D0	000D3	8\$: MOVL	#1, TST_SET	3841	
			0A	11	000D6	BRB	10\$	3778	
	67		02	FB	000D8	9\$: CALLS	#2, EC\$ERR_CNTL	3847	
	01		50	D1	000DB	CMPL	R0, #1		
			29	13	000DE	BEQL	13\$		
			3C	11	000E0	BRB	15\$		
		1C	AE	9F	000E2	10\$: PUSHAB	SPC	3853	
00000000G	00		01	FB	000E5	CALLS	#1, SC\$SKIP_COMMA		
			50	D5	000EC	TSTL	R0		
			03	13	000EE	BEQL	11\$		
		FF	55	31	000F0	BRW	3\$		
	03		54	E9	000F3	11\$: BLBC	NAM_SET, 12\$	3857	
	10		53	E8	000F6	BLBS	TST_SET, 13\$		
		001C81A4	8F	DD	000F9	12\$: PUSHL	#1868196	3860	
			52	DD	000FF	PUSHL	KYW_LINE		
	67		02	FB	00101	CALLS	#2, EC\$ERR_CNTL		
	01		50	D1	00104	CMPL	R0, #1		
			15	12	00107	BNEQ	15\$		
		28	AE	9F	00109	13\$: PUSHAB	BUFF	3868	
00000000G	00		01	FB	0010C	CALLS	#1, SI\$SKIP_IGNORED		
			50	D5	00113	TSTL	R0		
			03	13	00115	BEQL	14\$		


```
50          FF06 31 00117      BRW      1$
             01 D0 0011A 14$:    MOVL     #1, R0
             04 0011D          RET
             50 D4 0011E 15$:    CLRL     R0
             04 00120          RET
```

```
: 3874
: 3875
:
```

; Routine Size: 289 bytes, Routine Base: SOR\$RO_CODE + 0000

```
: 2689      3876 1
: 2690      3877 1 ! ELUDOM
: 2691      3878 1 !
: 2692      3879 1 END
: 2693      3880 0 ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
SOR\$RO_CODE	289	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	104	25	34	00:00.1
_\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	8	6	12	00:00.1
_\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI)

```
: 2694      L 3881 0 MODULE P4$PAS (IDENT = 'T03-038' %REQUIRE('SRC$:DEFS0') ) =
: 2695      3887 1 BEGIN
: 2696      3888 1 REQUIRE 'SRC$:DKS';
: 2697      4186 1 EXTERNAL ROUTINE
: 2698      4187 1 ERR_CNTL:      CA_LINKAGE,      ! Error control routine
: 2699      4188 1 SPC_ALLOC:    CA_LINKAGE,      ! Allocate storage from the work area
: 2700      4189 1 SKIP_IGNORED, ! Skip ignored text characters
```


P4\$PAS
T03-038

: 2701
: 2702

4190 1
4191 1

GET_NEXT_SPEC.
PARSE_KEY: CA_LINKAGE;

N 4
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 88
(22)

! Get pointer and length of next specification
! Parse key definitions

P5
T0


```

: 2704      4192 1 GLOBAL ROUTINE P4$PASS_FOUR(
: 2705      4193 1     START:      REF VECTOR[3],
: 2706      4194 1     CON_SYM:    REF CON_SYM_TAB,
: 2707      4195 1     P_CON_CNT:  REF VECTOR[1],
: 2708      4196 1     FCD_SYM:    REF SYM_TAB,
: 2709      4197 1     CDX_SYM:    REF SYM_TAB,
: 2710      4198 1     P_CDX_CNT:  REF VECTOR[1]
: 2711      4199 1     ): CA_LINKAGE =
: 2712      4200 2 BEGIN
: 2713      4201 2     -----
: 2714      4202 2     Pass 4:  parse all /KEY and /DATA specifications.
: 2715      4203 2     define the default record type in temporary
: 2716      4204 2     record definition table.
: 2717      4205 2     Build the KFT table
: 2718      4206 2     -----
: 2719      4207 2 BIND
: 2720      4208 2     CON_CNT = P_CON_CNT[0],
: 2721      4209 2     CDX_CNT = P_CDX_CNT[0];
: 2722      4210 2
: 2723      4211 2 LOCAL
: 2724      4212 2     KYW_TYPE,      ! Type of keyword
: 2725      4213 2     KYW_LINE,      ! Line number of keyword
: 2726      4214 2     BUFF:          VECTOR[3],
: 2727      4215 2     SPC:           VECTOR[3],      ! Next specification length/pointer/line
: 2728      4216 2     SUB_SPC:       VECTOR[3];      ! Sub specification length/pointer/line
: 2729      4217 2
: 2730      4218 2 ! Get the impure area
: 2731      4219 2 CA_AREA_( CA );
: 2732      4220 2
: 2733      4221 2     -----
: 2734      4222 2     Pass 4:  parse all /KEY and /DATA specifications.
: 2735      4223 2     define the default record type in temporary
: 2736      4224 2     record definition table.
: 2737      4225 2     Build the KFT table
: 2738      4226 2     -----
: 2739      4227 2 BEGIN                                     ! Pass 4
: 2740      4228 2 LOCAL
: 2741      4229 2     KEY_SET,      ! Has key been explicitly set
: 2742      4230 2     DATA_SET;    ! Has data been explicitly set
: 2743      4231 2
: 2744      4232 2 output_('pass 4');
: 2745      4233 2
: 2746      4234 2 ! Anchor key/data field table (KFT) at beginning of available work area
: 2747      4235 2 ! Set address of key/data field definition table
: 2748      4236 2 ! Key field table so far has no entries
: 2749      4237 2
: 2750      4238 2
: 2751      4239 2 %IF %BLISS(BLISS16) %THEN IF NOT .CA[CA_KEY_OVR] %FI
: 2752      4240 2 %IF %BLISS(BLISS32) %THEN IF .CA[CA_KFT_ADR] EQL 0 %FI
: 2753      4241 2 THEN
: 2754      4242 2     BEGIN
: 2755      4243 2     CA[CA_KFT_ADR] = ALLOC_(0);
: 2756      4244 2     CA[CA_KFT_SIZ] = 0;
: 2757      4245 2     END
: 2758      4246 2 ELSE
: 2759      4247 2     BEGIN
: 2760      4248 2     LOCAL
```



```
: 2761      4249  4      FROM_PTR;
: 2762      4250  4      !move the key table entries generated by command line /ke:
: 2763      4251  4      FROM_PTR = .CA[CA_KFT_ADR];
: 2764      4252  4      CA[CA_KFT_ADR] = ALLOC ( .CA[CA_KFT_SIZE] * KFT_UNIT );
: 2765      4253  4      CH$MOVE( .CA[CA_KFT_SIZE] * KFT_UNIT, .FROM_PTR, .CA[CA_KFT_ADR] );
: 2766      4254  4      END;
: 2767      4255  3
: 2768      4256  3
: 2769      4257  3      ! Initialize for pass 4
: 2770      4258  3      BUFF[LEN] = .START[LEN];
: 2771      4259  3      BUFF[PTR] = .START[PTR];
: 2772      4260  3      BUFF[LINE] = .START[LINE];
: 2773      4261  3
: 2774      4262  3      ! Neither key nor data has been explicitly given
: 2775      4263  3      KEY_SET = FALSE;
: 2776      4264  3      DATA_SET = FALSE;
: 2777      4265  3
: 2778      4266  3      ! Exit loop when there are no remaining specifications
: 2779      4267  4      WHILE TRUE DO
: 2780      4268  4          BEGIN
: 2781      4269  4              LABEL
: 2782      4270  4                  key:
: 2783      4271  4
: 2784      4272  4              KYW_LINE = .BUFF[LINE];
: 2785      4273  4              KYW_TYPE = GET_NEXT_SPEC( BUFF[0], SPC[0] );
: 2786      4274  4
: 2787      4275  4              ! Process the keyword
: 2788      4276  5      key:      BEGIN
: 2789      4277  5          SELECT .KYW_TYPE OF
: 2790      4278  5              SET
: 2791      4279  5
: 2792      4280  5              [KW_KEY, KW_DATA]: ! Make entry in the key/data field table (KFT)
: 2793      4281  6                  BEGIN
: 2794      4282  6                      LOCAL
: 2795      4283  6                          IS_DATA;
: 2796      4284  6
: 2797      4285  6                      IF .KYW_TYPE EQL KW_DATA
: 2798      4286  6                          THEN
: 2799      4287  7                              BEGIN
: 2800      4288  7                                  DATA_SET = TRUE;
: 2801      4289  7                                  IS_DATA = TRUE;
: 2802      4290  7                                  END
: 2803      4291  6                      ELSE
: 2804      4292  7                          BEGIN
: 2805      4293  7                              IF .CA[CA_KEY_OVR]
: 2806      4294  7                                  THEN
: 2807      4295  8                                  BEGIN
: 2808      4296  8                                      WRN ( SOR$_SPCOVR ); ! Warn of override
: 2809      4297  8                                      LEAVE key;
: 2810      4298  7                                  END;
: 2811      4299  7                                  KEY_SET = TRUE;
: 2812      4300  7                                  IS_DATA = FALSE;
: 2813      4301  6                                  END;
: 2814      4302  6
: 2815      4303  6              ! Parse the key or data
: 2816      4304  6              IF PARSE_KEY( SPC[0], .IS_DATA,
: 2817      4305  6                  CON_SYM[0, BASE_], CON_CNT,
```



```

: 2818      4306 6      FLD_SYM[0,BASE_],
: 2819      4307 6      CDX_SYM[0,BASE_], CDX_CNT ) EQL FAIL
: 2820      4308 6      THEN
: 2821      4309 6      ERR_( .SPC[ _LINE], SORS$_SPCIVK );    ! Invalid key or data field
: 2822      4310 6
: 2823      4311 5      END;
: 2824      4312 5
: 2825      4313 5      ! Bad specification keywords handled in pass 1
: 2826      4314 5      TES;
: 2827      4315 4      END;
: 2828      4316 4
: 2829      4317 4      ! Find the next specification, ignore comments and blanks
: 2830      4318 4      IF SKIP_IGNORED( BUFF[0] ) EQL FAIL THEN EXITLOOP;
: 2831      4319 4
: 2832      4320 3      END;
: 2833      4321 3
: 2834      4322 3      ! See if default key field needs to be defined
: 2835      4323 3      !
: 2836      4324 4      IF NOT ( .KEY_SET OR .CA[CA_KEY_OVR] )
: 2837      4325 3      THEN
: 2838      4326 4      BEGIN      ! Define default key entry
: 2839      4327 4      LOCAL
: 2840      4328 4      KFT:      REF KFT_TAB[];    ! Pointer to Key Field Table
: 2841      4329 4
: 2842      4330 4      KFT = ALLOC (KFT_UNIT);
: 2843      4331 4      IF .CA[CA_KFT_SIZ] GEQ KFT_MAX THEN ERR_( 0, SORS$_WKAREA );
: 2844      4332 4      CA[CA_KFT_SIZ] = .CA[CA_KFT_SIZ] + 1;
: 2845      4333 4
: 2846      4334 4      KFT[0, KFT_BUILD] = TRUE;
: 2847      4335 4      KFT[0, KFT_CONTINUE] = TRUE;
: 2848      4336 4      KFT[0, KFT_CONSTANT] = FALSE;
: 2849      4337 4      KFT[0, KFT_CONT_CD_X] = FALSE;
: 2850      4338 4      KFT[0, KFT_CONDX] = FALSE;
: 2851      4339 4      KFT[0, KFT_DESCEND] = FALSE;
: 2852      4340 4      KFT[0, KFT_DATA] = FALSE;
: 2853      4341 4      KFT[0, KFT_NDE_POS] = 0;
: 2854      4342 4      KFT[0, KFT_FDT_IDX] = 0;
: 2855      4343 4      KFT[0, KFT_TDT_IDX] = 0;
: 2856      4344 4
: 2857      4345 3      END;
: 2858      4346 3
: 2859      4347 3      ! See if default data field needs to be defined
: 2860      4348 3      !
: 2861      4349 3      IF NOT .DATA_SET
: 2862      4350 3      THEN
: 2863      4351 4      BEGIN      ! Define default data entry
: 2864      4352 4      LOCAL
: 2865      4353 4      KFT:      REF KFT_TAB[];    ! Pointer to Key Field Table
: 2866      4354 4
: 2867      4355 4      KFT = ALLOC (KFT_UNIT);
: 2868      4356 4      IF .CA[CA_KFT_SIZ] GEQ KFT_MAX THEN ERR_( 0, SORS$_WKAREA );
: 2869      4357 4      CA[CA_KFT_SIZ] = .CA[CA_KFT_SIZ] + 1;
: 2870      4358 4
: 2871      4359 4      KFT[0, KFT_BUILD] = TRUE;
: 2872      4360 4      KFT[0, KFT_CONTINUE] = TRUE;
: 2873      4361 4      KFT[0, KFT_DATA] = TRUE;
: 2874      4362 4      KFT[0, KFT_CONSTANT] = FALSE;
```



```

: 2875      4363 4      KFT[0, KFT-CONT-CDX] = FALSE;
: 2876      4364 4      KFT[0, KFT-COND] = FALSE;
: 2877      4365 4      KFT[0, KFT-DESCEND] = FALSE;
: 2878      4366 4      KFT[0, KFT-NDE-POS] = 0;
: 2879      4367 4      KFT[0, KFT-FDT-IDX] = 0;
: 2880      4368 4      KFT[0, KFT-TDT-IDX] = 0;
: 2881      4369 4
: 2882      4370 3      END;
: 2883      4371 3
: 2884      4372 3      ! Set continue bit off for last key field table entry
: 2885      4373 4      BEGIN
: 2886      4374 4      LOCAL KFT: REF KFT-TAB[];
: 2887      4375 4      KFT = .CA[CA-KFT-ADR];
: 2888      4376 4      KFT = KFT[.CA[CA-KFT-SIZ]-1, BASE_];
: 2889      4377 4      KFT[0, KFT-CONTINUE] = FALSE;
: 2890      4378 3      END;
: 2891      4379 3
: 2892      4380 2      END;
: 2893      4381 2      ! Pass 4
: 2894      4382 2      RETURN SUCCESS;
: 2895      4383 1      END;
```

```
.TITLE P4$PAS
.IDENT \T03-038\
```

```
.EXTRN EC$ERR-CNTL, SA$SPC-ALLOC
.EXTRN SI$SKIP-IGNORED
.EXTRN GN$GET-NEXT-SPEC
.EXTRN PK$PARSE-KEY
```

```
.PSECT SORSRO-CODE, NOWRT, SHR, PIC, 2
```

```
.ENTRY P4$PASS-FOUR, Save R2, R3, R4, R5, R6, R7, R8      ; 4192
MOVAB SA$SPC-ALLOC, R8
MOVAB EC$ERR-CNTL, R7
SUBL2 #36, SP
MOVAB 264(CA), R2      ; 4240
TSTL (R2)
BNEQ 1$
MOVL 296(CA), (R2)      ; 4243
CLRB 253(CA)      ; 4244
BRB 2$      ; 4240
MOVL (R2), FROM-PTR      ; 4251
MOVZBL 253(CA), R0      ; 4252
ASHL #3, R0, -(SP)
CALLS #1, SA$SPC-ALLOC
MOVL R0, (R2)
MOVZBL 253(CA), R0      ; 4253
MULL2 #8, R0
MOVC3 R0, (FROM-PTR), @264(CA)
MOVL START, R0      ; 4257
MOVQ (R0), BUFF
MOVL 8(R0), BUFF+8      ; 4259
CLRL KEY-SET      ; 4262
CLRL DATA-SET      ; 4263
MOVL BUFF+8, KYW-LINE      ; 4271
```

```

      01FC 00000
58 00000000G 00 9E 00002
57 00000000G 00 9E 00009
5E      24 C2 00010
52      0108 CB 9E 00013
      62 D5 00018
      0B 12 0001A
62      0128 CB D0 0001C
      00FD CB 94 00021
      20 11 00025
53      62 D0 00027 1$:
50      00FD CB 9A 0002A
7E      50      03 78 0002F
68      01 FB 00033
62      50 D0 00036
50      00FD CB 9A 00039
50      08 C4 0003E
0108 DB 63      50 28 00041
      50      04 AC D0 00047 2$:
      18 AE      60 7D 0004B
      20 AE      08 A0 D0 0004F
      54 D4 00054
      56 D4 00056
55      20 AE D0 00058 3$:
```


		OC	AE	9F	0005C	PUSHAB	SPC		4272
		1C	AE	9F	0005F	PUSHAB	BUFF		
00000000G	00		02	FB	00062	CALLS	#2, GNS\$GET_NEXT_SPEC		
	53		50	D0	00069	MOVL	R0, KYW_TYPE		
	2C		53	D1	0006C	CMPL	KYW_TYPE, #44		4280
			05	13	0006F	BEQL	4\$		
	32		53	D1	00071	CMPL	KYW_TYPE, #50		
			4D	12	00074	BNEQ	9\$		
	32		53	D1	00076	CMPL	KYW_TYPE, #50		4285
			08	12	00079	BNEQ	5\$		
	56		01	D0	0007B	MOVL	#1, DATA_SET		4288
	52		01	D0	0007E	MOVL	#1, IS_DATA		4289
OA	79	AB	14	11	00081	BRB	7\$		4285
		001C8160	01	E1	00083	BBC	#1, 121(CA), 6\$		4293
			8F	DD	00088	PUSHL	#1868128		4296
			55	DD	0008E	PUSHL	KYW_LINE		
			29	11	00090	BRB	8\$		
	54		01	D0	00092	MOVL	#1, KEY_SET		4299
			52	D4	00095	CLRL	IS_DATA		4300
	7E	14	AC	7D	00097	MOVQ	CDR_SYM, -(SP)		4307
	7E	OC	AC	7D	0009B	MOVQ	P_CON_CNT, -(SP)		
		08	AC	DD	0009F	PUSHL	CON_SYM		
			52	DD	000A2	PUSHL	IS_DATA		
		24	AE	9F	000A4	PUSHAB	SPC		4304
00000000G	00		07	FB	000A7	CALLS	#7, PK\$PARSE_KEY		4307
			50	D5	000AE	TSTL	R0		
		001C81AC	11	12	000B0	BNEQ	9\$		
		18	8F	DD	000B2	PUSHL	#1868204		4309
			AE	DD	000B8	PUSHL	SPC+8		
	67		02	FB	000BB	CALLS	#2, EC\$ERR_CNTL		
	01		50	D1	000BE	CMPL	R0, #1		
			6B	12	000C1	BNEQ	12\$		
		18	AE	9F	000C3	PUSHAB	BUFF		4318
00000000G	00		01	FB	000C6	CALLS	#1, SI\$SKIP_IGNORED		
			50	D5	000CD	TSTL	R0		
			87	12	000CF	BNEQ	3\$		
	39		54	E8	000D1	BLBS	KEY_SET, 11\$		4324
34	79	AB	01	E0	000D4	BBS	#1, 121(CA), 11\$		
			08	DD	000D9	PUSHL	#8		4330
	68		01	FB	000DB	CALLS	#1, SA\$SPC_ALLOC		
	53		50	D0	000DE	MOVL	R0, KFT		
	FF	8F	CB	91	000E1	CMPB	253(CA), #255		4331
		00FD	10	1F	000E7	BLSSU	10\$		
		001C815C	8F	DD	000E9	PUSHL	#1868124		
			7E	D4	000EF	CLRL	-(SP)		
	67		02	FB	000F1	CALLS	#2, EC\$ERR_CNTL		
	01		50	D1	000F4	CMPL	R0, #1		
			62	12	000F7	BNEQ	15\$		
		00FD	CB	96	000F9	INCB	253(CA)		4332
	50	03	A3	9E	000FD	MOVAB	3(KFT), R0		4334
	60		11	88	00101	BISB2	#17, (R0)		4335
	60	6E	8F	8A	00104	BICB2	#110, (R0)		4340
			63	B4	00108	CLRW	(KFT)		4341
		04	A3	B4	0010A	CLRW	4(KFT)		4342
	34		56	E8	0010D	BLBS	DATA_SET, 14\$		4349
			08	DD	00110	PUSHL	#8		4355
	68		01	FB	00112	CALLS	#1, SA\$SPC_ALLOC		

FF	52	00FD	50	D0	00115	MOVL	R0, KFT	
	8F		CB	91	00118	CMPB	253(CA), #255	4356
		001C815C	10	1F	0011E	BLSSU	13\$	
			8F	DD	00120	PUSHL	#1868124	
	67		7E	D4	00126	CLRL	-(SP)	
	01		02	FB	00128	CALLS	#2, EC\$ERR_CNTL	
			50	D1	0012B	CMPL	R0, #1	
			2B	12	0012E	BNEQ	15\$	
		00FD	CB	96	00130	INCB	253(CA)	4357
	50	03	A2	9E	00134	MOVAB	3(KFT), R0	4359
	60	51	8F	88	00138	BISB2	#81, (R0)	4361
	60		2E	8A	0013C	BICB2	#46, (R0)	4365
			62	B4	0013F	CLRW	(KFT)	4366
		04	A2	B4	00141	CLRW	4(KFT)	4367
	51	0108	CB	D0	00144	MOVL	264(CA), KFT	4375
	50	00FD	CB	9A	00149	MOVZBL	253(CA), R0	4376
	51	F8	A140	7E	0014E	MOVAQ	-8(KFT)[R0], KFT	
03	A1		01	8A	00153	BICB2	#1, 3(KFT)	4377
	50		01	D0	00157	MOVL	#1, R0	4382
				04	0015A	RET		
			50	D4	0015B	CLRL	R0	4383
				04	0015D	RET		

; Routine Size: 350 bytes, Routine Base: SOR\$RO_CODE + 0000

```

: 2896      4384 1
: 2897      4385 1 ! ELUDOM
: 2898      4386 1 !
: 2899      4387 1 END
: 2900      4388 0 ELUDOM

```

PSECT SUMMARY

Name	Bytes	Attributes
SOR\$RO_CODE	350	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	105	25	34	00:00.1
-\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	22	18	12	00:00.1
-\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LISS:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI
)

```
: 2901      L 4389 0 MODULE P5$PAS  (IDENT = 'T03-038' %REQUIRE('SRC$:DEFS0') ) =
: 2902      4395 1 BEGIN
: 2903      4396 1 REQUIRE 'SRC$:DKS';
: 2904      4694 1 EXTERNAL ROUTINE
: 2905      4695 1      ERR_CNTL:      CA_LINKAGE,      ! Error control routine
: 2906      4696 1      SPC_ALLOC:      CA_LINKAGE,      ! Allocate storage from the work area
: 2907      4697 1      SKIP_IGNORED,      ! Skip ignored text characters
: 2908      4698 1      SKIP_COMMA,      ! Skip commas and ignored stuff
: 2909      4699 1      GET_NEXT_SPEC,      ! Get pointer and length of next specification
: 2910      4700 1      GET_SUB_SPEC,      ! Get pointer and length of sub-specification
: 2911      4701 1      SEARCH_TABLE,      ! Search a table for a string
: 2912      4702 1      PARSE_KEY:      CA_LINKAGE,      ! Parse the key or data clause
: 2913      4703 1      CONV_CONSTANTS: CA_LINKAGE;      ! Convert constant symbol table to
```



```
2915 4704 1 GLOBAL ROUTINE P5$PASS_FIVE(
2916 4705 1     START:      REF VECTOR[3],
2917 4706 1     CON_SYM:   REF CON_SYM_TAB,
2918 4707 1     P_CON_CNT:  REF VECTOR[1],
2919 4708 1     FCD_SYM:    REF SYM_TAB,
2920 4709 1     CDX_SYM:    REF SYM_TAB,
2921 4710 1     P_CDX_CNT:  REF VECTOR[1],
2922 4711 1     TRDT:      REF TRDT_TAB
2923 4712 1 ):      CA_LINKAGE =
2924 4713 2 BEGIN
2925 4714 2 -----
2926 4715 2 Pass 5:  parse all /include and /omit specifications.
2927 4716 2          finish the temporary record definition table.
2928 4717 2          create the record definition table from the
2929 4718 2          temporary record definition table.
2930 4719 2          create the constant table.
2931 4720 2          Build the RDT table
2932 4721 2 -----
2933 4722 2 BIND
2934 4723 2     CON_CNT = P_CON_CNT[0],
2935 4724 2     CDX_CNT = P_CDX_CNT[0];
2936 4725 2
2937 4726 2 LOCAL
2938 4727 2     KYW_TYPE,           ! Type of keyword
2939 4728 2     SUB_KYW_TYPE,
2940 4729 2     KYW_LINE,         ! Line number of keyword
2941 4730 2     BUFF:             VECTOR[3],
2942 4731 2     SPC:               VECTOR[3],   ! Next specification length/pointer/line
2943 4732 2     SUB_SPC:          VECTOR[3],   ! Sub specification length/pointer/line
2944 4733 2     COL:               VECTOR[3],   ! Collating sequence specification
2945 4734 2     TRDT_CNT;
2946 4735 2
2947 4736 2
2948 4737 2 ! Get the impure area
2949 4738 2 CA_AREA_( CA );
2950 4739 2
2951 4740 2
2952 4741 2
2953 4742 2 -----
2954 4743 2 Pass 5:  parse all /include and /omit specifications.
2955 4744 2          finish the temporary record definition table.
2956 4745 2          create the record definition table from the
2957 4746 2          temporary record definition table.
2958 4747 2          create the constant table.
2959 4748 2          Build the RDT table
2960 4749 2 -----
2961 4750 2 BEGIN                                     ! Pass 5
2962 4751 2
2963 4752 2 output_('pass 5');
2964 4753 2
2965 4754 2 ! Initialize for pass 5
2966 4755 2 BUFF[LEN] = .START[LEN];
2967 4756 2 BUFF[PTR] = .START[PTR];
2968 4757 2 BUFF[LINE] = .START[LINE];
2969 4758 2
2970 4759 2 ! Initialize temporary record defn table count for this pass
2971 4760 2 TRDT_CNT = 0;
```



```

: 2972      4761 3
: 2973      4762 3
: 2974      4763 3      ! Exit loop when there are no remaining specifications
: 2975      4764 4      WHILE TRUE DO
: 2976      4765 4          BEGIN
: 2977      4766 4              LABEL
: 2978      4767 4                  include;
: 2979      4768 4
: 2980      4769 4          KYW_LINE = .BUFF[ LINE];
: 2981      4770 4          KYW_TYPE = GET_NEXT_SPEC( BUFF[0], SPC[0] );
: 2982      4771 4
: 2983      4772 4          ! Process the keyword
: 2984      4773 4          !
: 2985      4774 4          SELECTONE .KYW_TYPE OF
: 2986      4775 4              SET
: 2987      4776 4                  [KW_INCL, KW_OMIT]: ! Make entry in temporary record defn table
: 2988      4777 5          include:
: 2989      4778 5              BEGIN
: 2990      4779 5                  LITERAL
: 2991      4780 5                      CDX_SET = 0,
: 2992      4781 5                      KEY_SET = 1,
: 2993      4782 5                      DAT_SET = 2;
: 2994      4783 5              LOCAL
: 2995      4784 5                  FLAGS: BITVECTOR[3];
: 2996      4785 5              IF .TRDT_CNT GEQ RDT_MAX
: 2997      4786 6                  THEN
: 2998      4787 6                      BEGIN
: 2999      4788 6                          ERR ( .KYW_LINE, SORS_WKAREA );
: 3000      4789 6                          RETURN FAIC;
: 3001      4790 6                      END;
: 3002      4791 5              ! No key or data or condition found in include yet
: 3003      4792 5              FLAGS<0,%BPUNIT+%ALLOCATION(FLAGS),0> = 0;
: 3004      4793 5              ! Set the include bit and KFT index
: 3005      4794 5              IF .KYW_TYPE EQL KW_INCL
: 3006      4795 5                  THEN
: 3007      4796 5                      BEGIN
: 3008      4797 6                          TRDT[.TRDT_CNT, TRDT_INCLUDE] = TRUE;
: 3009      4798 6                          TRDT[.TRDT_CNT, TRDT_KFT_IDX] = .CA[CA_KFT_SIZE];
: 3010      4799 6                      END
: 3011      4800 6                  ELSE
: 3012      4801 5                      BEGIN
: 3013      4802 6                          TRDT[.TRDT_CNT, TRDT_INCLUDE] = FALSE;
: 3014      4803 6                      END;
: 3015      4804 5              ! Initialize the conditional bit to off
: 3016      4805 5              TRDT[.TRDT_CNT, TRDT_CONDX] = FALSE;
: 3017      4806 5
: 3018      4807 5              ! While there are more include or omit sub-specifications
: 3019      4808 5              WHILE .SPC[LEN] NEQ 0 DO
: 3020      4809 5                  BEGIN
: 3021      4810 5                      LABEL
: 3022      4811 6                          key;
: 3023      4812 6
: 3024      4813 6                      KYW_LINE = .SPC[ LINE];
: 3025      4814 6                      SUB_KYW_TYPE = GET_SUB_SPEC( SPC[0], SUB_SPC[0] );
: 3026      4815 6
: 3027      4816 6
: 3028      4817 6
```


3029 4818 6
3030 4819 6
3031 4820 7 key:
3032 4821 7
3033 4822 7
3034 4823 7
3035 4824 7
3036 4825 7
3037 4826 7
3038 4827 7
3039 4828 8
3040 4829 8
3041 4830 8
3042 4831 8
3043 4832 8
3044 4833 8
3045 4834 8
3046 4835 8
3047 4836 8
3048 4837 8
3049 4838 8
3050 4839 8
3051 4840 8
3052 4841 9
3053 4842 9
3054 4843 9
3055 4844 8
3056 4845 8
3057 4846 8
3058 4847 8
3059 4848 8
3060 4849 8
3061 4850 8
3062 4851 8
3063 4852 8
3064 4853 8
3065 4854 7
3066 4855 7
3067 4856 7
3068 4857 8
3069 4858 8
3070 4859 8
3071 4860 8
3072 4861 8
3073 4862 8
3074 4863 9
3075 4864 9
3076 4865 9
3077 4866 9
3078 4867 8
3079 4868 8
3080 4869 7
3081 4870 7
3082 4871 7
3083 4872 8
3084 4873 8
3085 4874 8

```
! Check the sub-specification
!
BEGIN
LOCAL
  IS_DATA;
SELECT .SUB_KYW_TYPE OF
  SET
  [KW_COND]:
    ! Condition
    BEGIN
    LOCAL
      INDEX,
      STATUS;
      ! Return status
      ! Search the condition symbol table for it
      STATUS = SEARCH_TABLE( SUB_SPC[0],
        INDEX, CDX_SYMC[0, BASE_], .CDX_CNT );
      ! Condition should not have been set and should have
      ! Non-zero length
      IF .STATUS EQL FAIL OR .FLAGS[CDX_SET]
      THEN
        BEGIN
          ERR ( .KYW_LINE, SOR$-SPCIVI ); ! Invalid include or omit
          LEAVE include;
        END;
      ! Condition has been set
      FLAGS[CDX_SET] = TRUE;
      ! Put the index in the temporary RDT
      TRDT[.TRDT_CNT, TRDT_CONDX] = TRUE;
      TRDT[.TRDT_CNT, TRDT_TDT_IDX] =
        .CDX_SYMC.INDEX, -SYM_INDEX];
    END;
  [KW_KEY]:
    BEGIN
      IS_DATA = FALSE;
      IF .CA[CA_KEY_OVR]
      THEN
        BEGIN
          WRN ( SOR$-SPCOVR ); ! Warn of override
          LEAVE key;
        END
      ELSE
        FLAGS[KEY_SET] = TRUE; ! Key has been given
      END;
  [KW_DATA]:
    BEGIN
      IS_DATA = TRUE;
```


3086 4875 8
3087 4876 8
3088 4877 8
3089 4878 7
3090 4879 7
3091 4880 7
3092 4881 8
3093 4882 8
3094 4883 8
3095 4884 8
3096 4885 8
3097 4886 9
3098 4887 9
3099 4888 9
3100 4889 8
3101 4890 8
3102 4891 8
3103 4892 8
3104 4893 8
3105 4894 8
3106 4895 8
3107 4896 8
3108 4897 9
3109 4898 9
3110 4899 9
3111 4900 8
3112 4901 8
3113 4902 7
3114 4903 7
3115 4904 7
3116 4905 8
3117 4906 8
3118 4907 8
3119 4908 7
3120 4909 7
3121 4910 7
3122 4911 6
3123 4912 6
3124 4913 6
3125 4914 5
3126 4915 5
3127 4916 5
3128 4917 5
3129 4918 5
3130 4919 5
3131 4920 5
3132 4921 5
3133 4922 5
3134 4923 5
3135 4924 5
3136 4925 5
3137 4926 5
3138 4927 6
3139 4928 5
3140 4929 5
3141 4930 5
3142 4931 5

```
        FLAGS[DAT_SET] = TRUE;      ! Data has been given
    END;
[KW_KEY, KW_DATA]:      ! Make entry in KFT table
    BEGIN
        ! If this is an omit then error
        IF .KYW_TYPE EQL KW_OMIT
        THEN
            BEGIN
                ERR ( .KYW_LINE, SORS$_SPCIVI ); ! Invalid or omit
                LEAVE include;
            END;

        ! Parse the key or data
        IF PARSE_KEY( SUB_SPC[0], .IS_DATA,
            CON_SYM[0,BASE_], CON_CNT,
            FLD_SYM[0,BASE_],
            CDX_SYM[0,BASE_], CDX_CNT ) EQL FAIL
        THEN
            BEGIN
                ERR ( .SUB_SPC[_LINE], SORS$_SPCIVI ); ! Invalid include or omit
                LEAVE include;
            END;
        END;

    [OTHERWISE]:      ! Bad include sub-specification
        BEGIN
            ERR ( .KYW_LINE, SORS$_SPCIVI ); ! Invalid include or omit
            LEAVE include;
        END;

    TES;
END;

IF SKIP_COMMA( SPC[0] ) EQL FAIL THEN EXITLOOP;
END;

! If no key or data set then put default key index in TRDT
IF NOT .FLAGS[KEY_SET] AND NOT .FLAGS[DAT_SET]
THEN
    TRDT[.TRDT_CNT, TRDT_KFT_IDX] = 0
ELSE
    IF .FLAGS[KEY_SET] AND .FLAGS[DAT_SET]
    THEN
        0
    ELSE
        IF NOT .FLAGS[KEY_SET]
        THEN
            ERR ( .KYW_LINE, SORS$_INCNOKEY )
        ELSE
            ERR ( .KYW_LINE, SORS$_INCNODATA );
```



```
3143 4932 5      ! Update the number of RDT table entries
3144 4933 5      TRDT_CNT = .TRDT_CNT + 1;
3145 4934 5
3146 4935 5      ! Turn continue bit off for last key field table entry
3147 4936 6      BEGIN
3148 4937 6      LOCAL KFT: REF KFT TAB[];
3149 4938 6      KFT = .CA[CA_KFT_ADR];
3150 4939 6      KFT[.CA[CA_KFT_SIZE]-1, KFT_CONTINUE] = FALSE;
3151 4940 6      END;
3152 4941 5
3153 4942 4      END;
3154 4943 4
3155 4944 4      TES;
3156 4945 4
3157 4946 4      ! Find the next specification, ignore comments and blanks
3158 4947 4      IF SKIP_IGNORED( BUFF[0] ) EQL FAIL THEN EXITLOOP;
3159 4948 4
3160 4949 3      END;
3161 4950 3
3162 4951 3      ! Add one more RDT entry for the fall-through/default case
3163 4952 3      IF .TRDT_CNT GEQ RDT_MAX
3164 4953 3      THEN
3165 4954 4      BEGIN
3166 4955 4      ERR ( .KYW LINE, SOR$_WKAREA );
3167 4956 4      RETURN FAIL;
3168 4957 3      END;
3169 4958 3      TRDT[.TRDT_CNT, TRDT_CONDX] = FALSE;
3170 4959 3      TRDT[.TRDT_CNT, TRDT_TDT_IDX] = 0;
3171 4960 3      TRDT[.TRDT_CNT, TRDT_KFT_IDX] = 0; ! ??? I think this is correct
3172 4961 4      TRDT[.TRDT_CNT, TRDT_INCLUDE] = (IF .TRDT_CNT EQL 0 THEN TRUE
3173 4962 3      ELSE NOT .TRDT[.TRDT_CNT-1, TRDT_INCLUDE]);
3174 4963 3      TRDT_CNT = .TRDT_CNT + 1;
3175 4964 3
3176 4965 3      ! Anchor address of record definition table to available work area
3177 4966 3      ! Set the address/size of the record definition table in common area
3178 4967 3      CA[CA_RDT_ADR] = ALLOC ( .TRDT_CNT * RDT_UNIT );
3179 4968 3      CA[CA_RDT_SIZE] = .TRDT_CNT;
3180 4969 3
3181 4970 3      ! Set up the record definition table
3182 4971 4      BEGIN
3183 4972 4      LOCAL
3184 4973 4      RDT: REF RDT TAB[]; ! Pointer to Record Defn Table
3185 4974 4      RDT = .CA[CA_RDT_ADR];
3186 4975 4      DECR IN FROM .TRDT_CNT-1 TO 0 DO
3187 4976 5      BEGIN
3188 4977 5      RDT[.IN, RDT_INCLUDE] = .TRDT[.IN, TRDT_INCLUDE];
3189 4978 5      RDT[.IN, RDT_CONDX] = .TRDT[.IN, TRDT_CONDX];
3190 4979 5      RDT[.IN, RDT_TDT_IDX] = .TRDT[.IN, TRDT_TDT_IDX];
3191 4980 5      RDT[.IN, RDT_KFT_IDX] = .TRDT[.IN, TRDT_KFT_IDX];
3192 4981 5      ! What about key length and data length fields???+++
3193 4982 4      END;
3194 4983 3      END;
3195 4984 3
3196 4985 3      ! Set the address/size of the constant field table in common area
3197 4986 3      CA[CA_CFT_ADR] = ALLOC ( .CON_CNT * CFT_UNIT );
3198 4987 3      CA[CA_CFT_SIZE] = .CON_CNT;
3199 4988 3      IF .CON_CNT GEQ CFT_MAX THEN ERR ( 0, SOR$_WKAREA );
```

```
! Convert the constant symbol table into the constant field table
IF CONV_CONSTANTS( .CA[CA_FDT_ADR], .CA[CA_CFT_ADR],
  CON_SYM[0,BASE_], .CON_CNT ) EQL FAIL
THEN
  BEGIN
    ERR ( 0, SOR$_SPCIVS );      ! Invalid specification
    RETURN FAIL;
  END;

! If we're at an odd boundary, allocate another byte
IF ALLOC_(0) THEN ALLOC_(1);

! Determine the lengths in bytes of the fields in the KFT table.
BEGIN
  BIND
    FDT = CA[CA_FDT_ADR]: REF FDT_TAB[];      ! Field definition table
    CFT = CA[CA_CFT_ADR]: REF CFT_TAB[];      ! Constant definition table
  LOCAL
    KFT1: REF KFT_TAB[],
    KFT2: REF KFT_TAB[],
    LEN;
  KFT1 = .CA[CA_KFT_ADR];
  KFT2 = .CA[CA_KFT_ADR];
  LEN = 0;
  DECR I FROM .CA[CA_KFT_SIZ]-1 TO 0 DO
    BEGIN
      LEN = MAXU(.LEN, KFT_UNITS_(KFT1));
      ! If this was the last key/data in the group, process the group
      IF NOT .KFT1[0,KFT_CONDX]
      THEN
        BEGIN
          WHILE KFT2[0,BASE_] LEQA KFT1[0,BASE_] DO
            BEGIN
              KFT2[0,KFT_NDE_SIZ] = .LEN;
              ??? Check for compatibility
              KFT2 = KFT2[1,BASE_];      ! Advance KFT2
            END;
          LEN = 0;                      ! Reinitialize for the next group
          KFT1 = KFT1[1,BASE_];          ! Advance KFT1
        END;
      END;
    END;
  END;

! Pass 5
RETURN SUCCESS;
END;
```

.TITLE P5\$PAS


```
.IDENT \T03-038\

.EXTRN ECSERR_CNTL, SASSPC_ALLOC
.EXTRN SI$SKIP_IGNORED
.EXTRN SC$SKIP_COMMA, GNS$GET_NEXT_SPEC
.EXTRN GSS$GET_SUB_SPEC
.EXTRN ST$SEARCH_TABLE
.EXTRN PK$PARSE_KEY, CC$CONV_CONSTANTS

.PSECT SORSRO_CODE, NOWRT, SHR, PIC, 2

.ENTRY P5$PASS_FIVE, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 4704
R10
MOVAB ECSERR_CNTL, R10
SUBL2 #52, SP
MOVL START, R0 4755
MOVQ (R0), BUFF
MOVL 8(R0), BUFF+8 4757
CLRL TRDT_CNT 4760
MOVL BUFF+8, KYW_LINE 4768
PUSHAB SPC 4769
PUSHAB BUFF
CALLS #2, GNS$GET_NEXT_SPEC
MOVL R0, KYW_TYPE
CMPL KYW_TYPE, #42 4775
BGEQ 3$
BRW 27$
CMPL KYW_TYPE, #43 2$:
BGTR 2$ 3$:
CMPL TRDT_CNT, #63 4784
BLEQ 4$
BRW 29$
CLRB FLAGS 4792
MULL3 #3, TRDT_CNT, R3 4798
ADDL2 TRDT, R3
CMPL KYW_TYPE, #42 4795
BNEQ 5$
BISB2 #1, (R3) 4798
MOVB 253(CA), 2(R3) 4799
BRB 6$ 4795
BICB2 #1, (R3) 4803
BICB2 #2, (R3) 4807
TSTL SPC 4810
BNEQ 8$
BRW 20$
MOVL SPC+8, KYW_LINE 4815
PUSHAB SUB_SPC 4816
PUSHAB SPC
CALLS #2, GSS$GET_SUB_SPEC
MOVL R0, SUB_KYW_TYPE
MOVL #1, R5 4824
CMPL SUB_KYW_TYPE, #51 4827
BNEQ 9$
CLRL R5
PUSHL @P CDX_CNT 4835
PUSHL CDX_SYM
PUSHAB INDEX 4834
```


00000000G	00	1C	AE 9F 00093	PUSHAB	SUB_SPC	4835
			04 FB 00096	CALLS	#4, ST\$SEARCH_TABLE	4839
			50 D5 0009D	TSTL	STATUS	
			55 13 0009F	BEQL	14\$	
	7D		56 E8 000A1	BLBS	FLAGS, 16\$	
	56		01 88 000A4	BISB2	#1, FLAGS	4847
	63		02 88 000A7	BISB2	#2, (R3)	4850
50	6E		06 C5 000AA	MULL3	#6, INDEX, R0	4852
	50	14	AC C0 000AE	ADDL2	CDX SYM, R0	
	01	05	A0 90 000B2	MOVB	5(R0), 1(R3)	
	A3		58 D1 000B7	CMPL	SUB_KYW_TYPE, #44	4856
	2C		1E 12 000BA	BNEQ	11\$	
			55 D4 000BC	CLRL	R5	
			57 D4 000BE	CLRL	IS_DATA	4859
12	79	AB	01 E1 000C0	BBC	#1, 121(CA), 10\$	4861
		001C8160	8F DD 000C5	PUSHL	#1868128	4864
			54 DD 000CB	PUSHL	KYW_LINE	
	6A		02 FB 000CD	CALLS	#2, EC\$ERR_CNTL	
	01		50 D1 000D0	CMPL	R0, #1	
			5F 13 000D3	BEQL	19\$	
			5A 11 000D5	BRB	18\$	
	56		02 88 000D7	BISB2	#2, FLAGS	4868
	32		58 D1 000DA	CMPL	SUB_KYW_TYPE, #50	4871
			08 12 000DD	BNEQ	12\$	
			55 D4 000DF	CLRL	R5	
	57		01 D0 000E1	MOVL	#1, IS_DATA	4874
	56		04 88 000E4	BISB2	#4, FLAGS	4876
	2C		58 D1 000E7	CMPL	SUB_KYW_TYPE, #44	4880
			05 13 000EA	BEQL	13\$	
	32		58 D1 000EC	CMPL	SUB_KYW_TYPE, #50	
			2D 12 000EF	BNEQ	15\$	
			55 D4 000F1	CLRL	R5	
	2B		59 D1 000F3	CMPL	KYW_TYPE, #43	4884
			29 13 000F6	BEQL	16\$	
	7E	14	AC 7D 000F8	MOVQ	CDX SYM, -(SP)	4895
	7E	0C	AC 7D 000FC	MOVQ	P CON CNT, -(SP)	
		08	AC DD 00100	PUSHL	CON SYM	
			57 DD 00103	PUSHL	IS_DATA	
		28	AE 9F 00105	PUSHAB	SUB_SPC	4892
00000000G	00		07 FB 00108	CALLS	#7, PK\$PARSE_KEY	4895
			50 D5 0010F	TSTL	R0	
			0B 12 00111	BNEQ	15\$	
		001C81B4	8F DD 00113	PUSHL	#1868212	4898
		1C	AE DD 00119	PUSHL	SUB_SPC+8	
			0B 11 0011C	BRB	17\$	
	13		55 E9 0011E	BLBC	R5, 19\$	4904
		001C81B4	8F DD 00121	PUSHL	#1868212	4906
			54 DD 00127	PUSHL	KYW_LINE	
	6A		02 FB 00129	CALLS	#2, EC\$ERR_CNTL	
	01		50 D1 0012C	CMPL	R0, #1	
			58 13 0012F	BEQL	27\$	
		01C5	31 00131	BRW	49\$	
			AE 9F 00134	PUSHAB	SPC	4913
00000000G	00	1C	01 FB 00137	CALLS	#1, SC\$SKIP_COMMA	
			50 D5 0013E	TSTL	R0	
			03 13 00140	BEQL	20\$	
			FF1F 31 00142	BRW	7\$	

0D	56	01	E0	00145	20\$:	BBS	#1, FLAGS, 22\$	4917
05	56	02	E0	00149		BBS	#2, FLAGS, 21\$	
		02	A3	94	0014D	CLRB	2(R3)	4919
			24	11	00150	BRB	26\$	
08	56	01	E1	00152	21\$:	BBC	#1, FLAGS, 23\$	4921
1C	56	02	E0	00156	22\$:	BBS	#2, FLAGS, 26\$	
08	56	01	E0	0015A		BBS	#1, FLAGS, 24\$	4927
	001C81BC	8F	DD	0015E	23\$:	PUSHL	#1868220	
		06	11	00164		BRB	25\$	
	001C81C4	8F	DD	00166	24\$:	PUSHL	#1868228	4929
		54	DD	0016C	25\$:	PUSHL	KYW_LINE	
6A		02	FB	0016E		CALLS	#2, EC\$ERR_CNTL	
01		50	D1	00171		CMPL	R0, #1	
		BB	12	00174		BNEQ	18\$	
		52	D6	00176	26\$:	INCL	TRDT_CNT	4933
51	0108	CB	D0	00178		MOVL	264(CA), KFT	4938
50	00FD	CB	9A	0017D		MOVZBL	253(CA), R0	4939
	FB	A140	7F	00182		PUSHAQ	-5(KFT)(R0)	
9E		01	8A	00186		BICB2	#1, @ (SP)+	
	28	AE	9F	00189	27\$:	PUSHAB	BUFF	4947
00000000G	00	01	FB	0018C		CALLS	#1, SI\$SKIP_IGNORED	
		50	D5	00193		TSTL	R0	
		03	13	00195		BEQL	28\$	
		FE81	31	00197		BRW	1\$	
	3F	52	D1	0019A	28\$:	CMPL	TRDT_CNT, #63	4952
		0B	15	0019D		BLEQ	30\$	
	001C815C	8F	DD	0019F	29\$:	PUSHL	#1868124	4955
		54	DD	001A5		PUSHL	KYW_LINE	
		00CF	31	001A7		BRW	36\$	
50	52	03	C5	001AA	30\$:	MULL3	#3, TRDT_CNT, R0	4958
51	50	AC	C1	001AE		ADDL3	TRDT, R0, R1	
	61	00FF0002	8F	CA	001B3	BICL2	#16711682, (R1)	4960
		52	D5	001BA		TSTL	TRDT_CNT	4961
		05	12	001BC		BNEQ	31\$	
	50	01	D0	001BE		MOVL	#1, R0	
		11	11	001C1		BRB	32\$	
50	52	03	C5	001C3	31\$:	MULL3	#3, TRDT_CNT, R0	4962
	50	AC	C0	001C7		ADDL2	TRDT, R0	
50	01	00	EF	001CB		EXTZV	#0, #1, -3(R0), R0	
61	50	50	D2	001D1		MCOML	R0, R0	
	01	50	F0	001D4	32\$:	INSV	R0, #0, #1, (R1)	4961
		52	D6	001D9		INCL	TRDT_CNT	4963
7E	52	06	C5	001DB		MULL3	#6, TRDT_CNT, -(SP)	4967
	00000000G	01	FB	001DF		CALLS	#1, SA\$SPC_ALLOC	
	0104	50	D0	001E6		MOVL	R0, 260(CA)	
	00FC	52	90	001EB		MOVB	TRDT_CNT, 252(CA)	4968
		54	CB	001F0		MOVL	260(CA), RDT	4974
		50	D0	001F5		MOVL	TRDT_CNT, IN	4977
		2A	11	001F8		BRB	34\$	
	52	06	C5	001FA	33\$:	MULL3	#6, IN, R2	
	53	52	C1	001FE		ADDL3	R2, RDT, R3	
	52	03	C5	00202		MULL3	#3, IN, R2	
	51	AC	C1	00206		ADDL3	TRDT, R2, R1	
63	01	61	F0	0020B		INSV	(R1), #0, #1, (R3)	
52	01	01	EF	00210		EXTZV	#1, #1, (R1), R2	4978
63	01	52	F0	00215		INSV	R2, #1, #1, (R3)	
		01	A1	90	0021A	MOVB	1(R1), 1(R3)	4979

04	A3	02	A1	90	0021F	MOV	2(R1), 4(R3)	4980
7E	0C		50	F4	00224	SOBGEQ	IN, 33\$	4975
00000000G	BC		06	C5	00227	MULL3	#6, @P_CON_CNT, -(SP)	4986
010C	00		01	FB	0022C	CALLS	#1, SASSPC-ALLOC	
00FE	CB	0C	50	D0	00233	MOVL	R0, 268(CA)	
000000FF	CB	0C	BC	90	00238	MOV	@P_CON_CNT, 254(CA)	4987
	8F	0C	BC	D1	0023E	CMPL	@P_CON_CNT, #255	4988
		001C815C	10	19	00246	BLSS	35\$	
			8F	DD	00248	PUSHL	#1868124	
	6A		7E	D4	0024E	CLRL	-(SP)	
01			02	FB	00250	CALLS	#2, EC\$ERR_CNTL	
			50	D1	00253	CMPL	R0, #1	
		0C	24	12	00256	BNEQ	37\$	
		08	BC	DD	00258	PUSHL	@P_CON_CNT	4992
		010C	AC	DD	0025B	PUSHL	CON_SYM	
		0110	CB	DD	0025E	PUSHL	268(CA)	
00000000G	00		CB	DD	00262	PUSHL	272(CA)	
		001C8184	04	FB	00266	CALLS	#4, CC\$CONV_CONSTANTS	
			50	D5	0026D	TSTL	R0	
			0D	12	0026F	BNEQ	38\$	
			8F	DD	00271	PUSHL	#1868164	4995
	6A		7E	D4	00277	CLRL	-(SP)	
			02	FB	00279	CALLS	#2, EC\$ERR_CNTL	
	09	0128	7B	11	0027C	BRB	49\$	4996
			CB	E9	0027E	BLBC	296(CA), 39\$	5001
			01	DD	00283	PUSHL	#1	
00000000G	00		01	FB	00285	CALLS	#1, SASSPC_ALLOC	
	52	0108	CB	D0	0028C	MOVL	264(CA), KFT1	5013
	54	0108	CB	D0	00291	MOVL	264(CA), KFT2	5014
			50	D4	00296	CLRL	LEN	5015
	55	00FD	CB	9A	00298	MOVZBL	253(CA), I	5016
			53	11	0029D	BRB	48\$	
	51	04	A2	9A	0029F	MOVZBL	4(KFT1), FDT_IX	5018
08	03		06	C4	002A3	MULL2	#6, R1	
			01	E1	002A6	BBC	#1, 3(KFT1), 41\$	
		010C	DB	41	9A	MOVZBL	@268(CA)[R1], R1	
			19	11	002B1	BRB	43\$	
		0110	CB	C0	002B3	ADDL2	272(CA), R1	41\$:
	51		61	91	002B8	CMPL	(R1), #21	
	15		0B	12	002BB	BNEQ	42\$	
		04	A1	3C	002BD	MOVZWL	4(R1), R1	
	51		02	C6	002C1	DIVL2	#2, R1	
			51	D6	002C4	INCL	R1	
			04	11	002C6	BRB	43\$	
	51	04	A1	3C	002C8	MOVZWL	4(R1), R1	42\$:
	53		50	D0	002CC	MOVL	LEN, R3	43\$:
	51		53	D1	002CF	CMPL	R3, R1	
			03	1E	002D2	BGEQU	44\$	
	53		51	D0	002D4	MOVL	R1, R3	
	50		53	D0	002D7	MOVL	R3, LEN	44\$:
10	03		03	E0	002DA	BBS	#3, 3(KFT1), 47\$	5022
			54	D1	002DF	CMPL	KFT2, KFT1	5025
			09	1A	002E2	BGTRU	46\$	
	06		50	B0	002E4	MOVW	LEN, 6(KFT2)	5027
			08	C0	002E8	ADDL2	#8, KFT2	5031
			F2	11	002EB	BRB	45\$	5025
			50	D4	002ED	CLRL	LEN	5033


```
52      08 C0 002EF 47$: ADDL2 #8, KFT1
AA      55 F4 002F2 48$: SOBGEQ 1, 40$
50      01 D0 002F5      MOVL #1, R0
          04 002F8      RET
          50 D4 002F9 49$: CLRL R0
          04 002FB      RET
```

```
: 5035
: 5016
: 5041
: 5042
:
```

; Routine Size: 764 bytes, Routine Base: SOR\$RO_CODE + 0000

```
: 3254      5043 1
: 3255      5044 1 ! ELUDOM
: 3256      5045 1 !
: 3257      5046 1 END
: 3258      5047 0 ELUDOM
```

PSECT SUMMARY

```
:
: Name          Bytes          Attributes
: SOR$RO_CODE    764 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)
```

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	108	26	34	00:00.1
-\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	35	29	12	00:00.1
-\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI)

```
: 3259      L 5048 0 MODULE ICT$IN (IDENT = 'T03-038' %REQUIRE('SRC$:DEFS0')) =
: 3260      5054 1 BEGIN
: 3261      5055 1 REQUIRE 'SRC$:DKS';
: 3262      5353 1 EXTERNAL ROUTINE
: 3263      5354 1 ERR CNTL: CA_LINKAGE, ! Error control routine
: 3264      5355 1 SOR$$DECM: CA_LINKAGE, ! Initialize for dec-multinational
```


ICT\$IN
T03-038

: 3265 5356 1
: 3266 5357 1
: 3267 5358 1
: 3268 5359 1
: 3269 5360 1
: 3270 5361 1
: 3271 5362 1
: 3272 5363 1
: 3273 5364 1

SORS\$EBCDIC: CA_LINKAGE,
COLL\$BASE,
COLL\$NEXT,
COLL\$MODIFY,
COLL\$FOLD,
COLL\$UPPER,
SKIP_COMMA,
GET_CHAR_CLAUSE,
GET_SUB_SPEC;

G 6
16-Sep-1984 00:53:54 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:54 [SORT32.SRC]SRTSPCBLI.B32;1

Page 107
(24)

: Initialize for ebcdic
: Define the base collating sequence
: Define the next character
: Make a modification
: Fold lower case to upper case
: Upper case comparison
: Skip commas and ignored stuff
: Get character clauses of various kinds
: Get pointer and length of sub-specification

SDI
T03

B7
44
5D
42
57
C7


```
3275 5365 1 GLOBAL ROUTINE INIT_CS TAB(
3276 5366 1     SUB_SPEC: REF VECTOR[3],
3277 5367 1     CD: REF VECTOR[2] ): CA_LINKAGE =
3278 5368 1
3279 5369 1 ++
3280 5370 1 Functional Description:
3281 5371 1     Initialize the Primary Table and Secondary Table.
3282 5372 1
3283 5373 1 Formal Parameters:
3284 5374 1
3285 5375 1     SUB_SPEC- The address of length/pointer/line defining the sequence
3286 5376 1     CD      - Length/address of the collating description work area
3287 5377 1
3288 5378 1 Implicit Inputs:
3289 5379 1
3290 5380 1 Implicit Outputs:
3291 5381 1
3292 5382 1 Routine Value:
3293 5383 1
3294 5384 1     SUCCESS
3295 5385 1     FAIL
3296 5386 1
3297 5387 1 Side Effects:
3298 5388 1
3299 5389 1 --
3300 5390 2 BEGIN
3301 5391 2
3302 5392 2 LOCAL
3303 5393 2     TEMP_SPC: VECTOR[3],      ! Temporary to hold length/pointer
3304 5394 2     DUMMY: VECTOR[3],
3305 5395 2     CLAUSE TYPE,
3306 5396 2     CS TYPE,      ! Collating sequence type returned from get_kyw_type
3307 5397 2     CH4: VECTOR[4,BYTE]; ! Characters returned from get_char_clause
3308 5398 2
3309 5399 2 LOCAL
3310 5400 2     KYW_LINE,
3311 5401 2     CHAR1: VECTOR[4,BYTE],
3312 5402 2     S;
3313 5403 2
3314 5404 2 ! Sets collating sequence type for sort-11 statistics
3315 5405 2 MACRO SET_STATS ( X ) =
3316 5406 2     %IF %BLISS(BLISS16)
3317 5407 2     %THEN
3318 5408 2         CA[ COM_COL_SEQ_ ] = X - SOR$_STASCI
3319 5409 2     %FI %;
3320 5410 2
3321 5411 2 CA_AREA_( CA );
3322 5412 2 output_('init_cs_tab');
3323 5413 2
3324 5414 2 CHAR1[0] = 1;
3325 5415 2 CHAR1[1] = 0;
3326 5416 2
3327 5417 2 ! Is there a sequence sub-clause?
3328 5418 2 IF .SUB_SPEC[LEN] EQL 0 THEN RETURN FAIL;
3329 5419 2
3330 5420 2 ! Save pointer and length since get_sub_spec modifies its input
3331 5421 2
```

```
3332 5422 2 TEMP_SPC[_PTR] = .SUB_SPEC[_PTR];
3333 5423 2 TEMP_SPC[_LEN] = .SUB_SPEC[_LEN];
3334 5424 2 TEMP_SPC[_LINE] = .SUB_SPEC[_LINE];
3335 5425 2
3336 5426 2 KYW_LINE = .SUB_SPEC[_LINE];
3337 5427 2
3338 5428 2 ! Get the collating sequence type
3339 5429 2 CS_TYPE = GET_SUB_SPEC( SUB_SPEC[0], DUMMY[0] );
3340 5430 2
3341 5431 2 ! Ascii, ebcdic, dec-multinational, or user_defined
3342 5432 2 SELECT .CS_TYPE OF
3343 5433 2 SET
3344 5434 2
3345 5435 2 [KW_ASC]: ! ASCII
3346 5436 2 BEGIN
3347 5437 2
3348 5438 2 ! Each character collates as its 8-bit character code
3349 5439 2 INCR I FROM %X'00' TO %X'FF' DO
3350 5440 2 BEGIN
3351 5441 2 CHAR1[2] = .I;
3352 5442 2 S = COLL$NEXT(CD[0], CHAR1);
3353 5443 2 IF_ERROR_( .S ) THEN (ERR_(.KYW_LINE, .S); RETURN FAIL);
3354 5444 2 END;
3355 5445 2
3356 5446 2 ! for sort-11 statistics, set collating sequence to ascii
3357 5447 2 SET_STATS_( SOR$_STASCII );
3358 5448 2
3359 5449 2 END;
3360 5450 2
3361 5451 2 [KW_EBC]: ! EBCDIC
3362 5452 2 BEGIN
3363 5453 2 S = SOR$_EBCDIC(CD[0]);
3364 5454 2 IF_ERROR_( .S ) THEN (ERR_(.KYW_LINE, .S); RETURN FAIL);
3365 5455 2
3366 5456 2 ! for sort-11 statistics, set collating sequence to ebcdic
3367 5457 2 SET_STATS_( SOR$_STEBCD );
3368 5458 2 END;
3369 5459 2
3370 5460 2 [KW_DEC]: ! DEC_Multinational
3371 5461 2 BEGIN
3372 5462 2 S = SOR$_DECM(CD[0]);
3373 5463 2 IF_ERROR_( .S ) THEN (ERR_(.KYW_LINE, .S); RETURN FAIL);
3374 5464 2
3375 5465 2 ! for sort-11 statistics, set collating sequence to multinational
3376 5466 2 SET_STATS_( SOR$_STDECM );
3377 5467 2 END;
3378 5468 2
3379 5469 2 [KW_UNKN]: ! User-defined base sequence
3380 5470 2 BEGIN
3381 5471 2
3382 5472 2 ! for sort-11 statistics, set collating sequence to user-defined
3383 5473 2 SET_STATS_( SOR$_STUSER );
3384 5474 2
3385 5475 2 ! Restore the length/address
3386 5476 2 SUB_SPEC[_PTR] = .TEMP_SPC[_PTR];
3387 5477 2 SUB_SPEC[_LEN] = .TEMP_SPC[_LEN];
3388 5478 2 SUB_SPEC[_LINE] = .TEMP_SPC[_LINE];
```



```
! While there are more user-defined sequence sub-clauses
WHILE TRUE DO
  BEGIN
```

```
! Get character clause
CLAUSE_TYPE = GET_CHAR_CLAUSE( SUB_SPEC[0], CH4[0] );
KYW_LINE = .SUB_SPEC[_LINE];
```

```
SELECTONE .CLAUSE_TYPE OF
  SET
```

```
    [GC_SINGLE, GC_S_TO_S]:      ! Single or range
    BEGIN
```

```
      ! Define entries in primary table
      CHAR1[0] = 1;
      INCRU I FROM .CH4[0] TO .CH4[2] DO
        BEGIN
          CHAR1[2] = .I;
          S = COLL$NEXT(CD[0], CHAR1);
          IF_ERROR_( .S ) THEN (ERR_(.KYW_LINE, .S); RETURN FAIL);
        END;
```

```
      END;
```

```
    [GC_DOUBLE]:
    BEGIN
      CHAR1[0] = 2;
      CHAR1[2] = .CH4[0];
      CHAR1[3] = .CH4[1];
      S = COLL$NEXT(CD[0], CHAR1);
      IF_ERROR_( .S ) THEN (ERR_(.KYW_LINE, .S); RETURN FAIL);
    END;
```

```
    [OTHERWISE]:                ! Error condition
    RETURN FAIL;
```

```
  TES;
```

```
  IF SKIP_COMMA( SUB_SPEC[0] ) EQL FAIL THEN EXITLOOP;
END;
```

```
END;
```

```
[OTHERWISE]:                  ! Misplaced keyword?
RETURN FAIL;
```

```
TES;
```

```
! Was there extraneous matter?
IF .SUB_SPEC[_LEN] NEQ 0 THEN RETURN FAIL;
```

```
RETURN SUCCESS;
END;
```

```
3389 5479 3
3390 5480 3
3391 5481 3
3392 5482 4
3393 5483 4
3394 5484 4
3395 5485 4
3396 5486 4
3397 5487 4
3398 5488 4
3399 5489 4
3400 5490 4
3401 5491 4
3402 5492 5
3403 5493 5
3404 5494 5
3405 5495 5
3406 5496 5
3407 5497 6
3408 5498 6
3409 5499 6
3410 5500 6
3411 5501 5
3412 5502 5
3413 5503 4
3414 5504 4
3415 5505 4
3416 5506 5
3417 5507 5
3418 5508 5
3419 5509 5
3420 5510 5
3421 5511 5
3422 5512 4
3423 5513 4
3424 5514 4
3425 5515 4
3426 5516 4
3427 5517 4
3428 5518 4
3429 5519 4
3430 5520 3
3431 5521 3
3432 5522 2
3433 5523 2
3434 5524 2
3435 5525 2
3436 5526 2
3437 5527 2
3438 5528 2
3439 5529 2
3440 5530 2
3441 5531 2
3442 5532 2
3443 5533 2
3444 5534 1
```

				.TITLE	ICT\$IN	
				.IDENT	\T03-038\	
				.EXTRN	ECSERR CNTL, SOR\$\$DECM	
				.EXTRN	SOR\$\$EBCDIC, COLL\$BASE	
				.EXTRN	COLL\$NEXT, COLL\$MODIFY	
				.EXTRN	COLL\$FOLD, COLL\$UPPER	
				.EXTRN	SC\$\$SKIP_COMMA, GCC\$GET_CHAR_CLAUSE	
				.EXTRN	GSS\$GET_SUB_SPEC	
				.PSECT	SOR\$RO_CODE, NOWRT, SHR, PIC, 2	
				.ENTRY	ICT\$INIT_CS_TAB, Save R2, R3, R4, R5, R6, R7, R8, -;	5365
					R9, R10	
				MOVAB	COLL\$NEXT, R10	
				SUBL2	#32, SP	
				MOVW	#1, CHAR1	5414
				MOVL	SUB_SPEC, R2	5418
				TSTL	(R2)	
				BNEQ	1\$	
				BRW	17\$	
				MOVQ	(R2), TEMP_SPC	5423
				MOVL	8(R2), TEMP_SPC+8	5424
				MOVL	8(R2), KYW_CINE	5426
				PUSHAB	DUMMY	5429
				PUSHL	R2	
				CALLS	#2, GSS\$GET_SUB_SPEC	
				MOVL	R0, CS_TYPE	
				MOVL	#1, R6	5432
				CMPL	CS_TYPE, #11	5435
				BNEQ	3\$	
				CLRL	R6	
				CLRL	I	5442
				MOVB	I, CHAR1+2	5441
				PUSHAB	CHAR1	5442
				PUSHL	CD	
				CALLS	#2, COLL\$NEXT	
				MOVL	R0, S	
				BLBC	S, 5\$	5443
				AOBLEQ	#255, I, 2\$	5439
				CMPL	CS_TYPE, #12	5451
				BNEQ	4\$	
				CLRL	R6	
				PUSHL	CD	5453
				CALLS	#1, SOR\$\$EBCDIC	
				MOVL	R0, S	
				BLBC	S, 11\$	5454
				CMPL	CS_TYPE, #13	5460
				BNEQ	6\$	
				CLRL	R6	
				PUSHL	CD	5462
				CALLS	#1, SOR\$\$DECM	
				MOVL	R0, S	
				BLBC	S, 14\$	5463
				CMPL	CS_TYPE, #65	5469
				BEQL	7\$	

				07FC 00000	
	5A	00000000G	00	9E	00002
	5E		20	C2	00009
04	AE		01	B0	0000C
	52		04	AC	00010
			62	D5	00014
			03	12	00016
			010E	31	00018
			62	7D	0001B 1\$:
14	AE		08	A2	0001F
1C	AE		08	A2	00024
	55		08	AE	00028
			52	DD	0002B
00000000G	00		02	FB	0002D
	59		50	D0	00034
	56		01	D0	00037
	0B		59	D1	0003A
			1F	12	0003D
			56	D4	0003F
			54	D4	00041
06	AE		54	90	00043 2\$:
			AE	9F	00047
			04	AC	DD 0004A
	6A		02	FB	0004D
	53		50	D0	00050
	33		53	E9	00053
E5	54	000000FF	8F	F3	00056
	0C		59	D1	0005E 3\$:
			12	12	00061
			56	D4	00063
			08	AC	DD 00065
00000000G	00		01	FB	00068
	53		50	D0	0006F
	67		53	E9	00072
	0D		59	D1	00075 4\$:
			12	12	00078
			56	D4	0007A
			08	AC	DD 0007C
00000000G	00		01	FB	0007F
	53		50	D0	00086
	78		53	E9	00089 5\$:
00000041	8F		59	D1	0008C 6\$:
			03	13	00093

			0086	31	00095	BRW	16\$	
			56	D4	00098	7\$: CLRL	R6	
08	62	14	AE	7D	0009A	MOVQ	TEMP-SPC, (R2)	5477
	A2	1C	AE	D0	0009E	MOVL	TEMP-SPC+8, 8(R2)	5478
		4004	8F	BB	000A3	8\$: PUSHR	#^M<R2, SP>	5485
00000000G	00		02	FB	000A7	CALLS	#2, GCC\$GET_CHAR_CLAUSE	
	57		50	D0	000AE	MOVL	R0, CLAUSE_TYPE	
	55	08	A2	D0	000B1	MOVL	8(R2), KYW_LINE	5486
	01		57	D1	000B5	CMPL	CLAUSE_TYPE, #1	5491
			05	13	000B8	BEQL	9\$	
	03		57	D1	000BA	CMPL	CLAUSE_TYPE, #3	
			29	12	000BD	BNEQ	13\$	
04	AE		01	90	000BF	9\$: MOVB	#1, CHAR1	5495
	58	02	AE	9A	000C3	MOVZBL	CH4+2, R8	5496
	54		6E	9A	000C7	MOVZBL	CH4, I	5499
			15	11	000CA	BRB	12\$	
06	AE		54	90	000CC	10\$: MOVB	I, CHAR1+2	5498
		04	AE	9F	000D0	PUSHAB	CHAR1	5499
		08	AC	DD	000D3	PUSHL	CD	
	6A		02	FB	000D6	CALLS	#2, COLL\$NEXT	
	53		50	D0	000D9	MOVL	R0, S	
	25		53	E9	000DC	11\$: BLBC	S, 14\$	5500
			54	D6	000DF	INCL	I	5496
	58		54	D1	000E1	12\$: CMPL	I, R8	
			E6	1B	000E4	BLEQU	10\$	
			29	11	000E6	BRB	15\$	5488
	02		57	D1	000E8	13\$: CMPL	CLAUSE_TYPE, #2	5505
			3C	12	000EB	BNEQ	17\$	
04	AE		02	90	000ED	MOVB	#2, CHAR1	5507
06	AE		6E	B0	000F1	MOVW	CH4, CHAR1+2	5508
		04	AE	9F	000F5	PUSHAB	CHAR1	5510
		08	AC	DD	000F8	PUSHL	CD	
	6A		02	FB	000FB	CALLS	#2, COLL\$NEXT	
	53		50	D0	000FE	MOVL	R0, S	
	0D		53	E8	00101	BLBS	S, 15\$	5511
			53	DD	00104	14\$: PUSHL	S	
			55	DD	00106	PUSHL	KYW_LINE	
00000000G	00		02	FB	00108	CALLS	#2, EC\$ERR_CNTL	
			18	11	0010F	BRB	17\$	
			52	DD	00111	15\$: PUSHL	R2	5519
00000000G	00		01	FB	00113	CALLS	#1, SC\$SKIP_COMMA	
			50	D5	0011A	TSTL	R0	
			85	12	0011C	BNEQ	8\$	
	08		56	E8	0011E	16\$: BLBS	R6, 17\$	5524
			62	D5	00121	TSTL	(R2)	5531
			04	12	00123	BNEQ	17\$	
	50		01	D0	00125	MOVL	#1, R0	5533
				04	00128	RET		
			50	D4	00129	17\$: CLRL	R0	5534
				04	0012B	RET		

; Routine Size: 300 bytes, Routine Base: SOR\$RO_CODE + 0000

```
3446 5535 1 GLOBAL ROUTINE DO_FOLD(CD: REF VECTOR[2]): CA_LINKAGE NOVALUE =
3447 5536 1
3448 5537 1 ++
3449 5538 1 Functional Description:
3450 5539 1
3451 5540 1 Fold the lower case characters onto the upper case.
3452 5541 1
3453 5542 1 Formal Parameters:
3454 5543 1
3455 5544 1 CD - Length/address of the collating description work area
3456 5545 1
3457 5546 1 Implicit Inputs:
3458 5547 1
3459 5548 1 Implicit Outputs:
3460 5549 1
3461 5550 1 Routine Value:
3462 5551 1
3463 5552 1 SUCCESS
3464 5553 1 FAIL
3465 5554 1
3466 5555 1 Side Effects:
3467 5556 1
3468 5557 1 --
3469 5558 2 BEGIN
3470 5559 2 LOCAL
3471 5560 2 LOWER: BITVECTOR[256],
3472 5561 2 CHAR1: VECTOR[4,BYTE],
3473 5562 2 CHAR2: VECTOR[4,BYTE],
3474 5563 2 S;
3475 5564 2
3476 5565 2 CA_AREA_( CA );
3477 5566 2
3478 5567 2 output_('do_fold');
3479 5568 2
3480 5569 2 CH$FILL(0, %ALLOCATION(LOWER), LOWER[0]);
3481 5570 2
3482 5571 2 (LOWER[XX'60'])<0,16,0> = XX'FFFE'; ! a - z
3483 5572 2 (LOWER[XX'70'])<0,16,0> = XX'07FF';
3484 5573 2
3485 5574 2 IF .CA[CA_BS_DECM]
3486 5575 2 THEN
3487 5576 2 BEGIN
3488 5577 2 (LOWER[XX'E0'])<0,16,0> = XX'FFFF'; ! a accent grave - y umlaut
3489 5578 2 (LOWER[XX'F0'])<0,16,0> = XX'3FFE';
3490 5579 2 END;
3491 5580 2
3492 5581 2 S = COLL$FOLD(CD[0], LOWER[0], XX'20');
3493 5582 2 IF_ERROR_(.S) THEN (ERR_(0,.S); RETURN);
3494 5583 2
3495 5584 2 RETURN;
3496 5585 2
3497 5586 1 END;
```


20	00	5E 6E	003C 00000 C2 00002 2C 00005 6E 0000A 8F D0 0000B 03 E1 00013 8F D0 00018 20 DD 00020 1\$: AE 9F 00022 AC DD 00025 03 FB 00028 50 E8 0002F 50 DD 00032 7E D4 00034 02 FB 00036 04 0003D 2\$:	.ENTRY DF\$DO FOLD, Save R2,R3,R4,R5 SUBL2 #32, SP MOVC5 #0, (SP), #0, #32, LOWER MOVL #134217726, LOWER+12 BBC #3, 121(CA), 1\$ MOVL #1073676287, LOWER+28 PUSHL #32 PUSHAB LOWER PUSHL CD CALLS #3, COLL\$FOLD BLBS S, 2\$ PUSHL S CLRL -(SP) CALLS #2, EC\$ERR_CNTL RET	: 5535 : 5569 : 5571 : 5574 : 5577 : 5581 : 5582 : 5586
	08	0C AE 07FFFFFFE 79 AB 1C AE 3FFEFFFFF			
		00000000G 00 OB			
		00000000G 00			

; Routine Size: 62 bytes, Routine Base: SOR\$RO_CODE + 012C

: 3498	5587	1
: 3499	5588	1
: 3500	5589	1 ! ELUDOM
: 3501	5590	1
: 3502	5591	1 END
: 3503	5592	0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
SOR\$RO_CODE	362	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	102	24	34	00:00.1
-\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	7	5	12	00:00.1
-\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LISS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI
)

```
: 3504      L 5593 0 MODULE SDECMS  (IDENT = 'T03-038' %REQUIRE('SRC$:DEFS0') ) =  
: 3505      5599 1 BEGIN  
: 3506      5600 1 REQUIRE 'SRC$:DKS';  
: 3507      5898 1 EXTERNAL ROUTINE  
: 3508      5899 1          COLL$NEXT,          ! Collate char after all others  
: 3509      5900 1          COLL$BASE,          ! Define the base collating sequence  
: 3510      5901 1          COLL$MODIFY,        ! Make a modification  
: 3511      5902 1          COLL$UPPER;        ! Upper case comparison  
: 3512      5903 1  
: 3513      5904 1 ! Define the successful status returned by these routines  
: 3514      5905 1 ! It should already be defined for bliss16 and should equal 0  
: 3515      5906 1  
: 3516      5907 1 %IF NOT %DECLARED(SS$_NORMAL) %THEN LITERAL SS$_NORMAL = 1; %FI  
: 3517      5908 1
```



```
3519 5909 1      ! Collating sequence tables for EBCDIC and DEC_MULTINATIONAL
3520 5910 1
3521 5911 1      ! Note that EBC_TAB has 128 entries and that DEC_TAB has 256 entries.
3522 5912 1
3523 5913 1      %IF %BLISS(BLISS32) %THEN
3524 5914 1
3525 5915 1      EXTERNAL
3526 5916 1          LIB$AB_ASC_EBC: ADDRESSING_MODE(GENERAL);
3527 5917 1      BIND
3528 5918 1          EBC_TAB = LIB$AB_ASC_EBC: VECTOR[,BYTE];
3529 5919 1
3530 5920 1      %ELSE
3531 5921 1
3532 5922 1      BIND
3533 5923 1          EBC_TAB = UPLIT BYTE(
3534 5924 1              'XX'00','XX'01','XX'02','XX'03','XX'1D','XX'19','XX'1A','XX'1B',      ! 00-07
3535 5925 1              'XX'0F','XX'04','XX'16','XX'06','XX'07','XX'08','XX'09','XX'0A',      ! 08-0F
3536 5926 1              'XX'0B','XX'0C','XX'0D','XX'0E','XX'1E','XX'1F','XX'1C','XX'17',      ! 10-17
3537 5927 1              'XX'10','XX'11','XX'20','XX'18','XX'12','XX'13','XX'14','XX'15',      ! 18-1F
3538 5928 1              'XX'21','XX'27','XX'3D','XX'39','XX'2A','XX'33','XX'28','XX'3B',      ! 20-27
3539 5929 1              'XX'25','XX'2C','XX'2B','XX'26','XX'32','XX'2F','XX'23','XX'30',      ! 28-2F
3540 5930 1              'XX'76','XX'77','XX'78','XX'79','XX'7A','XX'7B','XX'7C','XX'7D',      ! 30-37
3541 5931 1              'XX'7E','XX'7F','XX'38','XX'2D','XX'24','XX'3C','XX'35','XX'36',      ! 38-3F
3542 5932 1              'XX'3A','XX'5A','XX'5B','XX'5C','XX'5D','XX'5E','XX'5F','XX'60',      ! 40-47
3543 5933 1              'XX'61','XX'62','XX'64','XX'65','XX'66','XX'67','XX'68','XX'69',      ! 48-4F
3544 5934 1              'XX'6A','XX'6B','XX'6C','XX'6E','XX'6F','XX'70','XX'71','XX'72',      ! 50-57
3545 5935 1              'XX'73','XX'74','XX'75','XX'22','XX'6D','XX'29','XX'2E','XX'34',      ! 58-5F
3546 5936 1              'XX'37','XX'3E','XX'3F','XX'40','XX'41','XX'42','XX'43','XX'44',      ! 60-67
3547 5937 1              'XX'45','XX'46','XX'47','XX'48','XX'49','XX'4A','XX'4B','XX'4C',      ! 68-6F
3548 5938 1              'XX'4D','XX'4E','XX'4F','XX'51','XX'52','XX'53','XX'54','XX'55',      ! 70-77
3549 5939 1              'XX'56','XX'57','XX'58','XX'59','XX'31','XX'63','XX'50','XX'05',      ! 78-7F
3550 5940 1          ): VECTOR[,BYTE];
3551 5941 1      %FI
3552 5942 1
3553 5943 1      BIND
3554 5944 1          DEC_TAB = UPLIT BYTE(
3555 5945 1              'XX'01','XX'02','XX'03','XX'04','XX'05','XX'06','XX'07','XX'08',
3556 5946 1              'XX'09','XX'0A','XX'0B','XX'0C','XX'0D','XX'0E','XX'0F','XX'10',
3557 5947 1              'XX'11','XX'12','XX'13','XX'14','XX'15','XX'16','XX'17','XX'18',
3558 5948 1              'XX'19','XX'1A','XX'1B','XX'1C','XX'1D','XX'1E','XX'1F','XX'20',
3559 5949 1              'XX'21','XX'22','XX'23','XX'24','XX'25','XX'26','XX'27','XX'28',
3560 5950 1              'XX'29','XX'2A','XX'2B','XX'2C','XX'2D','XX'2E','XX'2F','XX'30',
3561 5951 1              'XX'31','XX'32','XX'33','XX'34','XX'35','XX'36','XX'37','XX'38',
3562 5952 1              'XX'39','XX'3A','XX'3B','XX'3C','XX'3D','XX'3E','XX'3F','XX'40',
3563 5953 1              'XX'41','XX'42','XX'43','XX'44','XX'45','XX'46','XX'47','XX'48',
3564 5954 1              'XX'49','XX'4A','XX'4B','XX'4C','XX'4D','XX'4E','XX'4F','XX'51',
3565 5955 1              'XX'52','XX'53','XX'54','XX'55','XX'56','XX'57','XX'58','XX'59',
3566 5956 1              'XX'5A','XX'5B','XX'5C','XX'60','XX'61','XX'62','XX'63','XX'64',
3567 5957 1              'XX'65','XX'42','XX'43','XX'44','XX'45','XX'46','XX'47','XX'48',
3568 5958 1              'XX'49','XX'4A','XX'4B','XX'4C','XX'4D','XX'4E','XX'4F','XX'51',
3569 5959 1              'XX'52','XX'53','XX'54','XX'55','XX'56','XX'57','XX'58','XX'59',
3570 5960 1              'XX'5A','XX'5B','XX'5C','XX'66','XX'67','XX'68','XX'69','XX'6A',
3571 5961 1              'XX'6B','XX'6C','XX'6D','XX'6E','XX'6F','XX'70','XX'71','XX'72',
3572 5962 1              'XX'73','XX'74','XX'75','XX'76','XX'77','XX'78','XX'79','XX'7A',
3573 5963 1              'XX'7B','XX'7C','XX'7D','XX'7E','XX'7F','XX'80','XX'81','XX'82',
3574 5964 1              'XX'83','XX'84','XX'85','XX'86','XX'87','XX'88','XX'89','XX'8A',
3575 5965 1              'XX'8B','XX'8C','XX'8D','XX'8E','XX'8F','XX'90','XX'91','XX'92',
```



```
3576 5966 1      '93', '94', '95', '96', '97', '98', '99', '9A',
3577 5967 1      '9B', '9C', '9D', '9E', '9F', 'A0', 'A1', 'A2',
3578 5968 1      'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'AA',
3579 5969 1      '42', '42', '42', '42', '42', '5F', '5D', '44',
3580 5970 1      '46', '46', '46', '46', '4A', '4A', '4A', '4A',
3581 5971 1      'AB', '50', '51', '51', '51', '51', '51', '51',
3582 5972 1      '5E', '57', '57', '57', '57', '5B', '5C', '55',
3583 5973 1      '42', '42', '42', '42', '42', '5F', '5D', '44',
3584 5974 1      '46', '46', '46', '46', '4A', '4A', '4A', '4A',
3585 5975 1      'AD', '50', '51', '51', '51', '51', '51', '51',
3586 5976 1      '5E', '57', '57', '57', '57', '5B', '5E', '5F',
3587 5977 1      : VECTOR[.BYTE];
3588 5978 1
3589 5979 1      BIND
3590 5980 1      DEC_UPP = PLIT BYTE(
3591 5981 1      '00', '01', '02', '03', '04', '05', '06', '07',
3592 5982 1      '08', '09', '0A', '0B', '0C', '0D', '0E', '0F',
3593 5983 1      '10', '11', '12', '13', '14', '15', '16', '17',
3594 5984 1      '18', '19', '1A', '1B', '1C', '1D', '1E', '1F',
3595 5985 1      '20', '21', '22', '23', '24', '25', '26', '27',
3596 5986 1      '28', '29', '2A', '2B', '2C', '2D', '2E', '2F',
3597 5987 1      '30', '31', '32', '33', '34', '35', '36', '37',
3598 5988 1      '38', '39', '3A', '3B', '3C', '3D', '3E', '3F',
3599 5989 1      '40', '41', '47', '48', '4A', '4B', '50', '51',
3600 5990 1      '52', '53', '58', '59', '5A', '5B', '5C', '5F',
3601 5991 1      '65', '66', '67', '69', '6A', '6B', '70', '71',
3602 5992 1      '72', '73', '75', '79', '7A', '7B', '7C', '7D',
3603 5993 1      '7E', '41', '47', '48', '4A', '4B', '50', '51',
3604 5994 1      '52', '53', '58', '59', '5A', '5B', '5C', '5F',
3605 5995 1      '65', '66', '67', '69', '6A', '6B', '70', '71',
3606 5996 1      '72', '73', '75', '7F', '80', '81', '82', '83',
3607 5997 1      '84', '85', '86', '87', '88', '89', '8A', '8B',
3608 5998 1      '8C', '8D', '8E', '8F', '90', '91', '92', '93',
3609 5999 1      '94', '95', '96', '97', '98', '99', '9A', '9B',
3610 6000 1      '9C', '9D', '9E', '9F', 'A0', 'A1', 'A2', 'A3',
3611 6001 1      'AC', 'AD', 'AE', 'AF', 'B0', 'B1', 'B2', 'B3',
3612 6002 1      'B4', 'B5', 'B6', 'B7', 'B8', 'B9', 'BA', 'BB',
3613 6003 1      'BC', 'BD', 'BE', 'BF', 'C0', 'C1', 'C2', 'C3',
3614 6004 1      '42', '43', '44', '45', '46', '78', '76', '49',
3615 6005 1      '4C', '4D', '4E', '4F', '54', '55', '56', '57',
3616 6006 1      'C4', '5D', '60', '61', '62', '63', '64', '5E',
3617 6007 1      '77', '6C', '6D', '6E', '6F', '74', 'C5', '68',
3618 6008 1      '42', '43', '44', '45', '46', '78', '76', '49',
3619 6009 1      '4C', '4D', '4E', '4F', '54', '55', '56', '57',
3620 6010 1      'C6', '5D', '60', '61', '62', '63', '64', '5E',
3621 6011 1      '77', '6C', '6D', '6E', '6F', '74', 'C7', 'C8',
3622 6012 1      : VECTOR[.BYTE];
```



```
3624 6013 1 GLOBAL ROUTINE SOR$$DECM(  
3625 6014 1     CD:      REF VECTOR[2]  
3626 6015 1     ): CA_LINKAGE =  
3627 6016 1  
3628 6017 1 ++  
3629 6018 1  
3630 6019 1 Functional Description:  
3631 6020 1  
3632 6021 1     Initializes a table for the dec multinational collating sequence.  
3633 6022 1  
3634 6023 1 Formal Parameters:  
3635 6024 1  
3636 6025 1     CD      - Length/address of the collating description work area  
3637 6026 1  
3638 6027 1 Implicit Inputs:  
3639 6028 1  
3640 6029 1     None.  
3641 6030 1  
3642 6031 1 Implicit Outputs:  
3643 6032 1  
3644 6033 1     None.  
3645 6034 1  
3646 6035 1 Routine Value:  
3647 6036 1  
3648 6037 1     0 on error.  
3649 6038 1     1 for sucess.  
3650 6039 1  
3651 6040 1 Side Effects:  
3652 6041 1  
3653 6042 1     None.  
3654 6043 1  
3655 6044 1 Notes:  
3656 6045 1  
3657 6046 1     After the call to this routine, the following calls still should  
3658 6047 1     be made: COLL$PAD, COLL$RESULT.  
3659 6048 1  
3660 6049 1 --  
3661 6050 2 BEGIN  
3662 6051 2 LOCAL  
3663 6052 2     S;  
3664 6053 2  
3665 6054 2     ! Get the impure area  
3666 6055 2     CA_AREA_( CA );  
3667 6056 2  
3668 6057 2     output_('sor$$decn');  
3669 6058 2  
3670 6059 2     S = COLL$BASE(CD[0], DEC TAB[0]);  
3671 6060 2     IF_ERROR( .S ) THEN RETURN .S;  
3672 6061 2     S = COLL$UPPER(CD[0], DEC UPP[0]);  
3673 6062 2     IF_ERROR_( .S ) THEN RETURN .S;  
3674 6063 2  
3675 6064 2     ! Small sharp s (DF), collates same as SS (53,53)  
3676 6065 2     ! OE dipthong (D7) collates same a OE (4F,45)  
3677 6066 2     !  
3678 6067 2     S = COLL$MODIFY(CD[0], UPLIT WORD(1,%X'DF'), UPLIT WORD(2,'SS'), 0);  
3679 6068 2     IF_ERROR( .S ) THEN RETURN .S;  
3680 6069 2     S = COLL$MODIFY(CD[0], UPLIT WORD(1,%X'D7'), UPLIT WORD(2,'OE'), 0);
```

```

: 3681      6070 2      IF_ERROR ( .S ) THEN RETURN .S;
: 3682      6071 2      S = COLL$MODIFY(CD[0], UPLIT WORD(1,XX'F7'), UPLIT WORD(2,'OE'), 0);
: 3683      6072 2      IF_ERROR_ ( .S ) THEN RETURN .S;
: 3684      6073 2
: 3685      6074 2      CA[CA_TIE_BREAK] = TRUE;
: 3686      6075 2      CA[CA_BS_DECM] = TRUE;
: 3687      6076 2
: 3688      6077 2      RETURN SS$_NORMAL;
: 3689      6078 2
: 3690      6079 1      END;
```

															.TITLE	SDECM\$																
															.IDENT	\T03-038\																
															.PSECT	SORSRO_CODE, NOWRT, SHR, PIC, 2																
OF	OE	OD	OC	OB	OA	O9	O8	O7	O6	O5	O4	O3	O2	O1	00000	P.AAA: .BYTE	1	2	3	4	5	6	7	8	9	10	11	12	-			
1E	1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10	0000F		13	14	15	16	17	18	19	20	21	22	-	-	-	:		
2D	2C	2B	2A	29	28	27	26	25	24	23	22	21	20	1F	0001E		23	24	25	26	27	28	29	30	31	32	-	-	-	:		
3C	3B	3A	39	38	37	36	35	34	33	32	31	30	2F	2E	0002D		33	34	35	36	37	38	39	40	41	42	-	-	-	:		
4B	4A	49	48	47	46	45	44	43	42	41	40	3F	3E	3D	0003C		43	44	45	46	47	48	49	50	51	52	-	-	-	:		
5B	5A	59	58	57	56	55	54	53	52	51	4F	4E	4D	4C	0004B		53	54	55	56	57	58	59	60	61	62	-	-	-	:		
49	48	47	46	45	44	43	42	41	40	3F	3E	3D	3C	3B	0005A		63	64	65	66	67	68	69	70	71	72	-	-	-	:		
59	58	57	56	55	54	53	52	51	4F	4E	4D	4C	4B	4A	00069		73	74	75	76	77	78	79	80	81	82	-	-	-	:		
71	70	6F	6E	6D	6C	6B	6A	69	68	67	66	65	64	63	00078		84	85	86	87	88	89	90	91	92	93	-	-	-	:		
80	7F	7E	7D	7C	7B	7A	79	78	77	76	75	74	73	72	00087		97	98	99	100	101	102	103	104	105	106	-	-	-	:		
8F	8E	8D	8C	8B	8A	89	88	87	86	85	84	83	82	81	00096		71	72	73	74	75	76	77	78	79	80	-	-	-	:		
9E	9D	9C	9B	9A	99	98	97	96	95	94	93	92	91	90	000A5		82	83	84	85	86	87	88	89	90	91	-	-	-	:		
42	42	42	AA	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	9F	000B4		92	102	103	104	105	106	107	108	-	-	-	-	-	:		
50	AB	4A	4A	4A	4A	46	46	46	46	44	5D	5F	42	42	000C3		109	110	111	112	113	114	115	116	-	-	-	-	-	:		
42	55	AC	5B	57	57	57	57	5E	51	51	51	51	51	51	000D2		117	118	119	120	121	122	123	124	-	-	-	-	-	:		
4A	4A	4A	4A	46	46	46	46	44	5D	5F	42	42	42	42	000E1		125	126	127	-128	-127	-126	-125	-	-	-	-	-	-	:		
AE	5B	57	57	57	57	5E	51	51	51	51	51	51	50	AD	000F0		-124	-123	-122	-121	-120	-119	-118	-	-	-	-	-	-	:		
														AF	000FF		-117	-116	-115	-114	-113	-112	-111	-	-	-	-	-	-	:		
																-110	-109	-108	-107	-106	-105	-104	-	-	-	-	-	-	-	:		
																-103	-102	-101	-100	-99	-98	-97	-	-	-	-	-	-	-	:		
																-96	-95	-94	-93	-92	-91	-90	-89	-	-	-	-	-	-	:		
																-88	-87	-86	-85	-84	-83	-82	-81	-	-	-	-	-	-	:		
																93	68	70	70	70	70	74	74	74	74	-	-	-	-	-	:	
																-85	80	81	81	81	81	81	81	81	81	-	-	-	-	-	:	
																87	87	87	91	-84	85	66	66	66	66	-	-	-	-	-	:	
																66	95	93	68	70	70	70	70	74	74	-	-	-	-	-	:	
																74	74	-83	80	81	81	81	81	81	81	-	-	-	-	-	:	
																94	87	87	87	87	91	-82	-81	-	-	-	-	-	-	-	:	
																64															:	
																00000040															:	
																00100															:	
OE	OD	OC	OB	OA	O9	O8	O7	O6	O5	O4	O3	O2	O1	00	00104	P.AAB: .LONG	0	1	2	3	4	5	6	7	8	9	10	11	12	-		
1D	1C	1B	1A	19	18	17	16	15	14	13	12	11	10	0F	00113	.BYTE	13	14	15	16	17	18	19	20	21	22	-	-	-	:		
2C	2B	2A	29	28	27	26	25	24	23	22	21	20	1F	1E	00122		23	24	25	26	27	28	29	30	31	32	-	-	-	:		
3B	3A	39	38	37	36	35	34	33	32	31	30	2F	2E	2D	00131		33	34	35	36	37	38	39	40	41	42	-	-	-	:		
5B	53	52	51	50	4B	4A	48	47	41	40	3F	3E	3D	3C	00140		43	44	45	46	47	48	49	50	51	52	-	-	-	:		
73	72	71	70	6B	6A	69	67	66	65	5F	5C	5B	5A	59	0014F		53	54	55	56	57	58	59	60	61	62	-	-	-	:		
52	51	50	4B	4A	48	47	41	7E	7D	7C	7B	7A	79	75	0015E		63	64	65	71	72	74	75	80	81	82	-	-	-	:		
71	70	6B	6A	69	67	66	65	5F	5C	5B	5A	59	58	53	0016D		83	88	89	90	91	92	95	101	102	-	-	-	-	-	:	
8A	89	88	87	86	85	84	83	82	81	80	7F	75	73	72	0017C		103	105	106	107	112	113	114	115	-	-	-	-	-	-	:	
99	98	97	96	95	94	93	92	91	90	8F	8E	8D	8C	8B	0018B		117	121	122	123	124	125	126	65	-	-	-	-	-	-	:	
AB	A7	A6	A5	A4	A3	A2	A1	A0	9F	9E	9D	9C	9B	9A	0019A		71	72	74	75	80	81	82	83	88	89	-	-	-	-	-	:

B7	B6	B5	B4	B3	B2	B1	B0	AF	AE	AD	AC	AB	AA	A9	001A9
44	43	42	C3	C2	C1	C0	BF	BE	BD	BC	BB	BA	B9	B8	001B8
5D	C4	57	56	55	54	4F	4E	4D	4C	49	76	78	46	45	001C7
42	68	C5	74	6F	6E	6D	6C	77	5E	64	63	62	61	60	001D6
57	56	55	54	4F	4E	4D	4C	49	76	78	46	45	44	43	001E5
C7	74	6F	6E	6D	6C	77	5E	64	63	62	61	60	5D	C6	001F4
													C8		00203

90	91	92	95	101	102	103	105	106	-	:
107	112	113	114	115	117	127	-128	-	-	:
-127	-126	-125	-124	-123	-122	-121	-	-	-	:
-120	-119	-118	-117	-116	-115	-114	-	-	-	:
-113	-112	-111	-110	-109	-108	-107	-	-	-	:
-106	-105	-104	-103	-102	-101	-100	-	-	-	:
-99	-98	-97	-96	-95	-94	-93	-92	-	-	:
-91	-90	-89	-88	-87	-86	-85	-84	-	-	:
-83	-82	-81	-80	-79	-78	-77	-76	-	-	:
-75	-74	-73	-72	-71	-70	-69	-68	-	-	:
-67	-66	-65	-64	-63	-62	-61	66	-	-	:
67	68	69	70	120	118	73	76	77	78	-
79	84	85	86	87	-60	93	96	97	98	-
99	100	94	119	108	109	110	111	-	-	:
116	-59	104	66	67	68	69	70	120	-	:
118	73	76	77	78	79	84	85	86	87	-
-58	93	96	97	98	99	100	94	119	-	:
108	109	110	111	116	-57	-56	-	-	-	:

00DF	0001	00204	P.AAC:	.WORD	1, 223
	0002	00208	P.AAD:	.WORD	2
	53	53		.ASCII	\SS\
00D7	0001	0020C	P.AAE:	.WORD	1, 215
	0002	00210	P.AAF:	.WORD	2
	45	4F		.ASCII	\OE\
00F7	0001	00214	P.AAG:	.WORD	1, 247
	0002	00218	P.AAH:	.WORD	2
	45	4F		.ASCII	\OE\

DEC_TAB=
DEC_UPP=

	P.AAA
	P.AAB
.EXTRN	COLL\$NEXT, COLL\$BASE
.EXTRN	COLL\$MODIFY, COLL\$UPPER
.EXTRN	LIB\$AB_ASC_EBC

		000C	00000	.ENTRY	SORT\$DECM, Save R2,R3	6013	
53	00000000G	00	9E	00002	MOVAB	COLL\$MODIFY, R3	
	FDD7	CF	9F	00009	PUSHAB	DEC_TAB	6059
52		04	AC	D0	MOVL	CD, R2	
			52	DD	PUSHL	R2	
00000000G	00		02	FB	CALLS	#2, COLL\$BASE	
	4C		50	E9	BLBC	S, 1\$	6060
	FEC7		CF	9F	PUSHAB	DEC_UPP	6061
			52	DD	PUSHL	R2	
00000000G	00		02	FB	CALLS	#2, COLL\$UPPER	
	3C		50	E9	BLBC	S, 1\$	6062
			7E	D4	CLRL	-(SP)	6067
	BA		AF	9F	PUSHAB	P.AAD	
	B3		AF	9F	PUSHAB	P.AAC	
			52	DD	PUSHL	R2	
63			04	FB	CALLS	#4, COLL\$MODIFY	
	2C		50	E9	BLBC	S, 1\$	6068
			7E	D4	CLRL	-(SP)	6069
	B2		AF	9F	PUSHAB	P.AAF	
	AB		AF	9F	PUSHAB	P.AAE	
			52	DD	PUSHL	R2	
63			04	FB	CALLS	#4, COLL\$MODIFY	
	1C		50	E9	BLBC	S, 1\$	6070

SDECM\$
T03-038

H 7
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 121
(28)

		AA	7E	D4	0004D	CLRL	-(SP)	:	6071
		A3	AF	9F	0004F	PUSHAB	P.AAH	:	
			AF	9F	00052	PUSHAB	P.AAG	:	
			52	DD	00055	PUSHL	R2	:	
	63		04	FB	00057	CALLS	#4, COLL\$MODIFY	:	
	OC		50	E9	0005A	BLBC	S, 1\$	:	6072
0080	CB		01	88	0005D	BISB2	#1, 128(CA)	:	6074
79	AB		08	88	00062	BISB2	#8, 121(CA)	:	6075
	50		01	D0	00066	MOVL	#1, R0	:	6077
			04	00069	1\$:	RET		:	6079

; Routine Size: 106 bytes, Routine Base: SOR\$R0_CODE + 021C


```
: 3692      6080 1 GLOBAL ROUTINE SOR$$EBCDIC( CD: REF VECTOR[2] ): CA_LINKAGE =
: 3693      6081 1
: 3694      6082 1 ++
: 3695      6083 1
: 3696      6084 1 Functional Description:
: 3697      6085 1
: 3698      6086 1     Initializes a table for the ebcdic collating sequence.
: 3699      6087 1
: 3700      6088 1 Formal Parameters:
: 3701      6089 1
: 3702      6090 1     CD      - Length/address of the collating description work area
: 3703      6091 1
: 3704      6092 1 Implicit Inputs:
: 3705      6093 1
: 3706      6094 1     None.
: 3707      6095 1
: 3708      6096 1 Implicit Outputs:
: 3709      6097 1
: 3710      6098 1     None.
: 3711      6099 1
: 3712      6100 1 Routine Value:
: 3713      6101 1
: 3714      6102 1     0 on error.
: 3715      6103 1     1 for sucess.
: 3716      6104 1
: 3717      6105 1 Side Effects:
: 3718      6106 1
: 3719      6107 1     None.
: 3720      6108 1
: 3721      6109 1 Notes:
: 3722      6110 1
: 3723      6111 1     After the call to this routine, the following calls still should
: 3724      6112 1     be made: COLL$PAD, COLL$RESULT.
: 3725      6113 1
: 3726      6114 1 --
: 3727      6115 2 BEGIN
: 3728      6116 2
: 3729      6117 2 LOCAL
: 3730      6118 2     CHAR1: VECTOR[ 4, BYTE ],
: 3731      6119 2     CHAR2: VECTOR[ 4, BYTE ],
: 3732      6120 2     S;
: 3733      6121 2
: 3734      6122 2 ! Get the impure area
: 3735      6123 2 CA_AREA_( CA );
: 3736      6124 2
: 3737      6125 2 output_( 'sor$$ebcdic' );
: 3738      6126 2
: 3739      6127 2 CHAR1[ 0 ] = 1;
: 3740      6128 2 CHAR1[ 1 ] = 0;
: 3741      6129 2 CHAR2[ 0 ] = 1;
: 3742      6130 2 CHAR2[ 1 ] = 0;
: 3743      6131 2
: 3744      6132 2 ! First half of the primary table
: 3745      6133 2 S = COLL$BASE( CD[0], EBC TAB[0] );
: 3746      6134 2 IF_ERROR_( .S ) THEN RETURN .S;
: 3747      6135 2
: 3748      6136 2 ! The second half collates the same as the SUB character (%X'1A')
```

```
: 3749      6137 2      CHAR2[ 2 ] = %X'1A';
: 3750      6138 2
: 3751      6139 2      INCR I FROM %X'80' TO %X'FF' DO
: 3752      6140 2      BEGIN
: 3753      6141 2      CHAR1[ 2 ] = .I;
: 3754      6142 2
: 3755      6143 2      S = COLL$MODIFY( CD[0], CHAR1, CHAR2, 0);
: 3756      6144 2      IF_ERROR_( .S ) THEN RETURN .S;
: 3757      6145 2
: 3758      6146 2      END;
: 3759      6147 2
: 3760      6148 2      CHAR1[2] = %X'FF';
: 3761      6149 2      S = COLL$NEXT( CD[0], CHAR1 );
: 3762      6150 2      IF_ERROR_( .S ) THEN RETURN .S;
: 3763      6151 2
: 3764      6152 2      RETURN SS$_NORMAL;
: 3765      6153 1      END;
```

			0004 00000	.ENTRY	SOR\$\$EBCDIC, Save R2	: 6080
			08 C2 00002	SUBL2	#8, SP	
	04	5E	01 B0 00005	MOVW	#1, CHAR1	: 6127
		6E	01 B0 0C009	MOVW	#1, CHAR2	: 6129
			00 9F 0000C	PUSHAB	EBC_TAB	: 6133
		00000000G	AC DD 00012	PUSHL	CD	
		04	02 FB 00015	CALLS	#2, COLL\$BASE	
	00000000G	00	50 E9 0001C	BLBC	S, 2\$	: 6134
		40	1A 90 0001F	MOVB	#26, CHAR2+2	: 6137
	02	AE	8F 9A 00023	MOVZBL	#128, I	: 6139
		52	52 90 00027	MOVB	I, CHAR1+2	: 6141
	06	AE	7E D4 0002B	CLRL	-(SP)	: 6143
			04 AE 9F 0002D	PUSHAB	CHAR2	
			0C AE 9F 00030	PUSHAB	CHAR1	
			04 AC DD 00033	PUSHL	CD	
	00000000G	00	04 FB 00036	CALLS	#4, COLL\$MODIFY	
		1F	50 E9 0003D	BLBC	S, 2\$	: 6144
DF		52	8F F3 00040	AOBLEQ	#255, I, 1\$	: 6139
	06	AE	01 8E 00048	MNEGB	#1, CHAR1+2	: 6148
			04 AE 9F 0004C	PUSHAB	CHAR1	: 6149
			04 AC DD 0004F	PUSHL	CD	
	00000000G	00	02 FB 00052	CALLS	#2, COLL\$NEXT	
		03	50 E9 00059	BLBC	S, 2\$	: 6150
		50	01 D0 0005C	MOVL	#1, R0	: 6152
			04 0005F	RET		: 6153

; Routine Size: 96 bytes, Routine Base: SOR\$RO_CODE + 0286

```
: 3766      6154 1
: 3767      6155 1
: 3768      6156 1 ! ELUDOM
: 3769      6157 1 !
: 3770      6158 1 END
: 3771      6159 0 ELUDOM
```


PSECT SUMMARY

Name	Bytes	Attributes
SOR\$RO_CODE	742	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	102	24	34	00:00.1
_\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	8	6	12	00:00.1
_\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.0

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI)

```

: 3772      L 6160 0 MODULE GFSS$GE  (IDENT = 'T03-038' %REQUIRE('SRC$:DEFS0') ) =
: 3773      6166 1 BEGIN
: 3774      6167 1 REQUIRE 'SRC$:DKS';

```

```
: 3776 6465 1 GLOBAL ROUTINE GET_FILE_SPEC( DESC: REF VECTOR[3], STRING: REF VECTOR[3] ) =
: 3777 6466 1
: 3778 6467 1 ++
: 3779 6468 1 Functional Description:
: 3780 6469 1
: 3781 6470 1     Get the file specification in DESC.
: 3782 6471 1
: 3783 6472 1 Formal Parameters:
: 3784 6473 1
: 3785 6474 1     DESC    - Address of length/address to search
: 3786 6475 1     STRING  - Address of stored length/address of string.
: 3787 6476 1
: 3788 6477 1 Implicit Inputs:
: 3789 6478 1
: 3790 6479 1     None.
: 3791 6480 1
: 3792 6481 1 Implicit Outputs:
: 3793 6482 1
: 3794 6483 1     None.
: 3795 6484 1
: 3796 6485 1 Routine Value:
: 3797 6486 1
: 3798 6487 1     SUCCESS
: 3799 6488 1     FAIL
: 3800 6489 1
: 3801 6490 1 Side Effects:
: 3802 6491 1
: 3803 6492 1 --
: 3804 6493 2 BEGIN
: 3805 6494 2 LOCAL
: 3806 6495 2     QUOTED,      ! Are we in a quote
: 3807 6496 2     BRAKET,    ! Are we in brackets
: 3808 6497 2     LENGTH;    ! Length of the found string
: 3809 6498 2
: 3810 6499 2 output_('get_file_spec');
: 3811 6500 2
: 3812 6501 2 QUOTED = FALSE;
: 3813 6502 2 BRAKET = FALSE;
: 3814 6503 2
: 3815 6504 2 ! Initialize length
: 3816 6505 2 LENGTH = 0;
: 3817 6506 2
: 3818 6507 2 ! Set pointer to string
: 3819 6508 2 STRING[PTR] = .DESC[PTR];
: 3820 6509 2 STRING[LINE] = .DESC[LINE];
: 3821 6510 2
: 3822 6511 2 ! Find the string
: 3823 6512 2 WHILE TRUE DO
: 3824 6513 2 BEGIN
: 3825 6514 2
: 3826 6515 2     IF .DESC[LEN] LEQ 0 THEN EXITLOOP;           ! Out of input
: 3827 6516 2
: 3828 6517 2     ! Look at the next character
: 3829 6518 2     SELECTONE CH$RCHAR(.DESC[PTR]) OF
: 3830 6519 2     SET
: 3831 6520 2
: 3832 6521 3     [C_NULL,C_SPACE,C_TAB,C_EXCLAM,C_L_PAREN,C_R_PAREN,C_EQUAL]:
```



```
3833 6522 3      IF NOT .QUOTED THEN EXITLOOP;    ! Are we inside of a quote?
3834 6523      [C_COMMA]:
3835 6524      IF NOT .QUOTED THEN
3836 6525      IF NOT .BRACKET THEN EXITLOOP;    ! Inside of [nnn,nnn]?
3837 6526
3838 6527
3839 6528      [C_LBRACK,C_LESS]:
3840 6529      IF NOT .QUOTED
3841 6530      THEN
3842 6531      BEGIN
3843 6532      IF .BRACKET THEN RETURN FAIL;
3844 6533      BRACKET = TRUE;
3845 6534      END;
3846 6535
3847 6536      [C_RBRACK,C_GREATER]:
3848 6537      IF NOT .QUOTED
3849 6538      THEN
3850 6539      BEGIN
3851 6540      IF NOT .BRACKET THEN RETURN FAIL;
3852 6541      BRACKET = FALSE;
3853 6542      END;
3854 6543
3855 6544      [C_QUOTE]:    ! Character '"'
3856 6545      QUOTED = NOT .QUOTED;
3857 6546
3858 6547      [OTHERWISE]:
3859 6548      0;
3860 6549
3861 6550      TES;
3862 6551
3863 6552      ! Update the values
3864 6553      DESC[_PTR] = .DESC[_PTR] + 1;
3865 6554      DESC[_LEN] = .DESC[_LEN] - 1;
3866 6555
3867 6556      LENGTH = .LENGTH + 1;
3868 6557
3869 6558      END;
3870 6559
3871 6560      IF .BRACKET OR .QUOTED THEN RETURN FAIL;
3872 6561
3873 6562      IF (STRING[_LEN] = .LENGTH) EQL 0
3874 6563      THEN
3875 6564      RETURN FAIL
3876 6565      ELSE
3877 6566      RETURN SUCCESS;
3878 6567
3879 6568      END;
```

.TITLE GFSSGE
.IDENT \T03-038\

003C 00000

.PSECT SORSRO_CODE,NOWRT, SHR, PIC,2
.ENTRY GFSSGET_FILE_SPEC, Save R2,R3,R4,R5

: 6465

		53	7C	00002	CLRQ	QUOTED	:	6501
		55	D4	00004	CLRL	LENGTH	:	6505
04	51	04	AC	7D	00006	MOVQ	DESC, R1	6508
	A2	04	A1	7D	0000A	MOVQ	4(R1), 4(R2)	
		61	D5	0000F	1\$: TSTL	(R1)	:	6515
		72	15	00011	BLEQ	12\$	:	
	50	04	B1	9A	00013	MOVZBL	24(R1), R0	6518
		1E	13	00017	BEQL	4\$	:	6521
	09	50	91	00019	CMPB	R0, #9	:	
		19	13	0001C	BEQL	4\$	:	
	20	50	91	0001E	CMPB	R0, #32	:	
		05	1F	00021	BLSSU	2\$	:	
	21	50	91	00023	CMPB	R0, #33	:	
		0F	1B	00026	BLEQU	4\$	:	
	28	50	91	00028	2\$: CMPB	R0, #40	:	
		05	1F	0002B	BLSSU	3\$	:	
	29	50	91	0002D	CMPB	R0, #41	:	
		05	1B	00030	BLEQU	4\$	:	
	3D	50	91	00032	3\$: CMPB	R0, #61	:	
		05	12	00035	BNEQ	5\$	:	
	42	53	E8	00037	4\$: BLBS	QUOTED, 11\$	:	6522
		49	11	0003A	BRB	12\$	:	
	2C	50	91	0003C	5\$: CMPB	R0, #44	:	6524
		08	12	0003F	BNEQ	6\$	:	
	38	53	E8	00041	BLBS	QUOTED, 11\$	:	6525
	35	54	E8	00044	BLBS	BRACKET, 11\$	:	6526
		3C	11	00047	BRB	12\$	:	
	3C	50	91	00049	6\$: CMPB	R0, #60	:	6528
		06	13	0004C	BEQL	7\$	:	
5B	8F	50	91	0004E	CMPB	R0, #91	:	
		0B	12	00052	BNEQ	8\$	:	
	25	53	E8	00054	7\$: BLBS	QUOTED, 11\$	:	6529
	3A	54	E8	00057	BLBS	BRACKET, 13\$	:	6532
	54	01	D0	0005A	MOVL	#1, BRACKET	:	6533
		1D	11	0005D	BRB	11\$	:	6529
	3E	50	91	0005F	8\$: CMPB	R0, #62	:	6536
		06	13	00062	BEQL	9\$	:	
5D	8F	50	91	00064	CMPB	R0, #93	:	
		0A	12	00068	BNEQ	10\$	:	
		53	E8	0006A	9\$: BLBS	QUOTED, 11\$	:	6537
	0F	54	E9	0006D	BLBC	BRACKET, 13\$	:	6540
	24	54	D4	00070	CLRL	BRACKET	:	6541
		08	11	00072	BRB	11\$	:	6537
	22	50	91	00074	10\$: CMPB	R0, #34	:	6544
		03	12	00077	BNEQ	11\$	:	
	53	53	D2	00079	MCOML	QUOTED, QUOTED	:	6545
		04	A1	D6	0007C	11\$: INCL	4(R1)	6553
		61	D7	0007F	DECL	(R1)	:	6554
		55	D6	00081	INCL	LENGTH	:	6556
		8A	11	00083	BRB	1\$	:	6512
	0C	54	E8	00085	12\$: BLBS	BRACKET, 13\$	:	6560
	09	53	E8	00088	BLBS	QUOTED, 13\$	:	
	62	55	D0	0008B	MOVL	LENGTH, (R2)	:	6562
		04	13	0008E	BEQL	13\$	:	
	50	01	D0	00090	MOVL	#1, R0	:	6566
		04	04	00093	RET		:	
		50	D4	00094	13\$: CLRL	R0	:	6568

04 00096

RET

: Routine Size: 151 bytes, Routine Base: SOR\$RO_CODE + 0000

```
: 3880      6569  1
: 3881      6570  1
: 3882      6571  1 ! ELUDOM
: 3883      6572  1 !
: 3884      6573  1 END
: 3885      6574  0 ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
SOR\$RO_CODE	151	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	4	0	34	00:00.1
-\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	17	14	12	00:00.1
-\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.0

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI)

```
: 3886      L 6575  0 MODULE PC$PAR (IDENT = 'T03-038' %REQUIRE('SRC$:DEFS0')) =
: 3887      6581  1 BEGIN
: 3888      6582  1 REQUIRE 'SRC$:DKS';
: 3889      6880  1 EXTERNAL ROUTINE
: 3890      6881  1 ERR_CNTL: CA_LINKAGE, ! Error control routine
: 3891      6882  1 SPC_ALLOC: CA_LINKAGE, ! Allocate storage from the work area
: 3892      6883  1 SKIP_COMMA, ! Skip commas and ignored stuff
: 3893      6884  1 GET_SUB_SPEC, ! Get pointer and length of sub-specification
: 3894      6885  1 INIT_CS_TAB: CA_LINKAGE, ! Initialize primary table
: 3895      6886  1 DO_FOLD: CA_LINKAGE NOVALUE, ! Fold lower case characters to upper
```

PC\$PAR
T03-038

C 8
16-Sep-1984 00:53:54 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:54 [SORT32.SRC]SRTSPCBL1.B32;1

Page 129
(30)

: 3896	6887	1	PARSE_MOD:	CA_LINKAGE,	: Parse a modification clause
: 3897	6888	1	PARSE_IGN:	CA_LINKAGE;	: Parse an ignore clause
: 3898	6889	1	EXTERNAL ROUTINE		
: 3899	6890	1	COLL\$INIT,		: Initialize collating sequence
: 3900	6891	1	COLL\$FOLD,		: Fold lower case to upper case
: 3901	6892	1	COLL\$TIE_BREAK,		: Indicate tie-breaking
: 3902	6893	1	COLL\$PAD,		: Indicate the pad character
: 3903	6894	1	COLL\$UPPER,		: Upper case comparison
: 3904	6895	1	COLL\$RESULT;		: Build the final tables


```
3906 6896 1 GLOBAL ROUTINE PARSE COLL(
3907 6897 1     DESC: REF VECTOR[3] ): CA_LINKAGE =
3908 6898 1
3909 6899 1 !++
3910 6900 1 Functional Description:
3911 6901 1
3912 6902 1     Parse the collating sequence specification.
3913 6903 1
3914 6904 1 Formal Parameters:
3915 6905 1
3916 6906 1     DESC    Address of length/address of the text.
3917 6907 1
3918 6908 1 Implicit Inputs:
3919 6909 1
3920 6910 1     None.
3921 6911 1
3922 6912 1 Implicit Outputs:
3923 6913 1
3924 6914 1     None.
3925 6915 1
3926 6916 1 Routine Value:
3927 6917 1
3928 6918 1     SUCCESS/FAIL
3929 6919 1
3930 6920 1 Side Effects:
3931 6921 1
3932 6922 1 --
3933 6923 2 BEGIN
3934 6924 2 LOCAL
3935 6925 2     COLL_DESC: VECTOR[2],
3936 6926 2     SUB_DESC:  VECTOR[3],
3937 6927 2     PAD_CHAR:  VECTOR[4,BYTE],
3938 6928 2     KYW_LINE,
3939 6929 2     S;
3940 6930 2
3941 6931 2 LOCAL
3942 6932 2     MODIFIED;
3943 6933 2
3944 6934 2 ! Get the impure data area
3945 6935 2 CA_AREA_( CA );
3946 6936 2
3947 6937 2 output_('parse_coll');
3948 6938 2
3949 6939 2 COLL_DESC[0] = .CA[CA_WRK_END]-.CA[CA_WRK_ADR];
3950 6940 2 COLL_DESC[1] = ALLOC_10;
3951 6941 2
3952 6942 2 S = COLL$INIT(COLL_DESC[0]);
3953 6943 2 IF_ERROR_( .S ) THEN (ERR_(.DESC[_LINE], .S); RETURN FAIL);
3954 6944 2
3955 6945 2 KYW_LINE = .DESC[_LINE];
3956 6946 2
3957 6947 2 ! The first clause must be SEQUENCE=...
3958 6948 2 !
3959 6949 2 IF GET_SUB_SPEC( DESC[0], SUB_DESC[0] ) EQL KW_SEQU
3960 6950 2 THEN
3961 6951 2 BEGIN
3962 6952 2     ! Call routine to initialize collating tables
```

```
3963 6953 3 IF INIT_CS_TAB( SUB_DESC[0], COLL_DESC[0] ) EQL FAIL
3964 6954 3 THEN
3965 6955 4 BEGIN
3966 6956 4 ERR ( .KYW_LINE, SOR$_SPCIVC ); ! Invalid collating sequence
3967 6957 4 RETURN FAIL;
3968 6958 4 END;
3969 6959 3 IF SUB_DESC[_LEN] GTR 0
3970 6960 3 THEN
3971 6961 4 BEGIN
3972 6962 4 ERR ( .KYW_LINE, SOR$_SPCIVC ); ! Invalid collating sequence
3973 6963 4 RETURN FAIL;
3974 6964 4 END;
3975 6965 3 END
3976 6966 3 ELSE
3977 6967 3 BEGIN
3978 6968 3 ERR ( .KYW_LINE, SOR$_SPCIVC ); ! Invalid collating sequence
3979 6969 3 RETURN FAIL;
3980 6970 3 END;
3981 6971 3
3982 6972 3 ! While there are more collating sequence sub-qualifiers
3983 6973 3 !
3984 6974 3 WHILE SKIP_COMMA( DESC[0] ) NEQ FAIL DO
3985 6975 3 BEGIN
3986 6976 3
3987 6977 3 KYW_LINE = .DESC[_LINE];
3988 6978 3
3989 6979 3 MODIFIED = FALSE;
3990 6980 3
3991 6981 3 SELECTONE GET_SUB_SPEC( DESC[0], SUB_DESC[0] ) OF
3992 6982 3 SET
3993 6983 3
3994 6984 3 [KW_MODF]:
3995 6985 3 BEGIN
3996 6986 3 ! Call routine to modify collating sequence tables
3997 6987 3 IF PARSE_MOD( SUB_DESC[0], COLL_DESC[0] ) EQL FAIL
3998 6988 3 THEN
3999 6989 3 BEGIN
4000 6990 3 ERR ( .KYW_LINE, SOR$_SPCIVC ); ! Invalid collating sequence
4001 6991 3 RETURN FAIL;
4002 6992 3 END;
4003 6993 3
4004 6994 3 MODIFIED = TRUE;
4005 6995 3
4006 6996 3 END;
4007 6997 3
4008 6998 3 [KW_IGNORE]: ! Specify ignored characters
4009 6999 3 BEGIN
4010 7000 3 ! Call routine to parse the ignore clause
4011 7001 3 IF PARSE_IGNORE( SUB_DESC[0], COLL_DESC[0] ) EQL FAIL
4012 7002 3 THEN
4013 7003 3 BEGIN
4014 7004 3 ERR ( .KYW_LINE, SOR$_SPCIVC ); ! Invalid collating sequence
4015 7005 3 RETURN FAIL;
4016 7006 3 END;
4017 7007 3
4018 7008 3 MODIFIED = TRUE;
4019 7009 3
```



```

: 4020      7010      4
: 4021      7011      4
: 4022      7012      4
: 4023      7013      4
: 4024      7014      4
: 4025      7015      4
: 4026      7016      4
: 4027      7017      4
: 4028      7018      4
: 4029      7019      4
: 4030      7020      4
: 4031      7021      4
: 4032      7022      4
: 4033      7023      4
: 4034      7024      4
: 4035      7025      4
: 4036      7026      4
: 4037      7027      4
: 4038      7028      4
: 4039      7029      4
: 4040      7030      4
: 4041      7031      4
: 4042      7032      4
: 4043      7033      4
: 4044      7034      4
: 4045      7035      4
: 4046      7036      4
: 4047      7037      4
: 4048      7038      4
: 4049      7039      4
: 4050      7040      4
: 4051      7041      4
: 4052      7042      4
: 4053      7043      4
: 4054      7044      4
: 4055      7045      4
: 4056      7046      4
: 4057      7047      4
: 4058      7048      4
: 4059      7049      4
: 4060      7050      4
: 4061      7051      4
: 4062      7052      4
: 4063      7053      4
: 4064      7054      4
: 4065      7055      4
: 4066      7056      4
: 4067      7057      4
: 4068      7058      4
: 4069      7059      4
: 4070      U 7060      4
: 4071      U 7061      4
: 4072      U 7062      4
: 4073      U 7063      4
: 4074      7064      4
: 4075      7065      4
: 4076      7066      4

      END;

[KW_FOLD]:
      BEGIN
      ! Perform the folding when found
      ! Note that if modifications are made to a dec_multi
      ! Sequence, folding may have to be done after mods
      DO_FOLD(COLL_DESC[0]);
      MODIFIED = TRUE;
      END;

[KW_TIEB]:      ! Tie-breaking
      BEGIN
      CA[CA_TIE_BREAK] = TRUE;
      MODIFIED = TRUE;
      END;

[KW_NOTI]:      ! No tie_breaking
      BEGIN
      CA[CA_TIE_BREAK] = FALSE;
      MODIFIED = TRUE;
      END;

[OTHERWISE]:      ! Bad collate sub-specification
      BEGIN
      ERR ( .KYW_LINE, SORS_SPCIVC ); ! Invalid collating sequence
      RETURN FAIL;
      END;

      TES;

      IF .SUB_DESC[LEN] GTR 0
      THEN
      BEGIN
      ERR ( .KYW_LINE, SORS_SPCIVC ); ! Invalid collating sequence
      RETURN FAIL;
      END;

      END;

      %IF %BLISS(BLISS16) %THEN
      IF .MODIFIED EQL ON
      THEN
      CA[ COM_COL_SEQ_ ] = .CA[ COM_COL_SEQ_ ] + 4;
      %FI

      IF .DESC[LEN] GTR 0
```

```

: 4077      7067 2
: 4078      7068
: 4079      7069
: 4080      7070
: 4081      7071
: 4082      7072
: 4083      7073
: 4084      7074
: 4085      7075
: 4086      7076
: 4087      7077
: 4088      7078
: 4089      7079
: 4090      7080
: 4091      7081
: 4092      7082
: 4093      7083
: 4094      7084
: 4095      7085
: 4096      7086
: 4097      7087
: 4098      7088
: 4099      7089
: 4100      7090
: 4101      7091
: 4102      7092
: 4103      7093
: 4104      7094
: 4105      7095
: 4106      7096
: 4107      7097
: 4108      7098
: 4109      7099
: 4110      7100
: 4111      7101
: 4112      7102
: 4113      7103
: 4114      7104
: 4115      7105
: 4116      7106
: 4117      7107
: 4118      7108
: 4119      7109 1

THEN
  BEGIN
    ERR ( .KYW_LINE, SORS$_SPCIVC ); ! Invalid collating sequence
    RETURN FAIL;
  END;

IF .CA[CA_TIE_BREAK]
THEN
  BEGIN
    ! Indicate tie-breaking.
    ! ALWAYS reverse the sense of the CMPC comparison.
    S = COLL$TIE_BREAK(COLL_DESC[0], .CA[CA_BS_DECM]);
    S = COLL$TIE_BREAK(COLL_DESC[0], TRUE);
  END
ELSE
  BEGIN
    ! Don't use the upper table if there is no tie-breaking.
    ! Currently, this is only needed if CA_BS_DECM is set.
    S = COLL$UPPER(COLL_DESC[0], 0);
  END;
IF_ERROR_( .S ) THEN RETURN (ERR_(.DESC[_LINE], .S); RETURN FAIL);

PAD_CHAR[0] = 1;
PAD_CHAR[1] = 0;
PAD_CHAR[2] = .CA[CA_PAD];
S = COLL$PAD(COLL_DESC[0], PAD_CHAR);
IF_ERROR_( .S ) THEN RETURN (ERR_(.DESC[_LINE], .S); RETURN FAIL);

S = COLL$RESULT(COLL_DESC[0], COLL_DESC[0]);
IF_ERROR_( .S ) THEN RETURN (ERR_(.DESC[_LINE], .S); RETURN FAIL);

! Store the address and length in the common area
!
CA[CA_ST_SIZ] = .COLL_DESC[0];
CA[CA_ST_ADR] = ALLOC_(.COLL_DESC[0]);

RETURN SUCCESS;
END;
```

```

.TITLE PC$PAR
.IDENT \T03-038\

.EXTRN EC$ERR CNTL, SASSPC ALLOC
.EXTRN SC$SKIP COMMA, GSS$GET_SUB_SPEC
.EXTRN ICT$INIT CS_TAB
.EXTRN DF$DO FOLD, PM$PARSE MOD
.EXTRN PI$PARSE IGN, COLL$INIT
.EXTRN COLL$FOLD, COLL$TIE_BREAK
.EXTRN COLL$PAD, COLL$UPPER
.EXTRN COLL$RESULT
```


				.PSECT	SOR\$RO_CODE,NOWRT, SHR, PIC,2	
			01FC 00000	.ENTRY	PC\$PARSE COLL, Save R2,R3,R4,R5,R6,R7,R8	6896
		58 00000000G	00 9E 00002	MOVAB	GSS\$GET_SUB_SPEC, R8	
		5E	18 C2 00009	SUBL2	#24, SP	
10	AE	012C	CB C3 0000C	SUBL3	296(CA), 300(CA), COLL_DESC	6939
		14	CB D0 00015	MOVL	296(CA), COLL_DESC+4	6940
			AE 9F 0001B	PUSHAB	COLL_DESC	6942
	00000000G		00 01 FB 0001E	CALLS	#1, COLL\$INIT	
			56 50 D0 00025	MOVL	R0, S	
		0C	56 E8 00028	BLBS	S, 1\$	6943
			56 DD 0002B	PUSHL	S	
		50	04 AC D0 0002D	MOVL	DESC, R0	
			08 A0 DD 00031	PUSHL	8(R0)	
			0108 31 00034	BRW	16\$	
		54	04 AC D0 00037	MOVL	DESC, R4	6945
		57	08 A4 9E 0003B	MOVAB	8(R4), R7	
		53	67 D0 0003F	MOVL	(R7), KYW_LINE	
			04 AE 9F 00042	PUSHAB	SUB_DESC	6949
			54 DD 00045	PUSHL	R4	
		68	02 FB 00047	CALLS	#2, GSS\$GET_SUB_SPEC	
		0A	50 D1 0004A	CMPL	R0, #10	
			7F 12 0004D	BNEQ	8\$	
			10 AE 9F 0004F	PUSHAB	COLL_DESC	6953
			08 AE 9F 00052	PUSHAB	SUB_DESC	
	00000000G	00	02 FB 00055	CALLS	#2, ICT\$INIT_CS_TAB	
			50 D5 0005C	TSTL	R0	
			7E 13 0005E	BEQL	11\$	
			04 AE D5 00060	TSTL	SUB_DESC	6959
			79 14 00063	BGTR	11\$	
			54 DD 00065	PUSHL	R4	6975
	00000000G	00	01 FB 00067	CALLS	#1, SC\$SKIP_COMMA	
			50 D5 0006E	TSTL	R0	
			68 13 00070	BEQL	10\$	
		53	67 D0 00072	MOVL	(R7), KYW_LINE	6978
			55 D4 00075	CLRL	MODIFIED	6980
			04 AE 9F 00077	PUSHAB	SUB_DESC	6982
			54 DD 0007A	PUSHL	R4	
		68	02 FB 0007C	CALLS	#2, GSS\$GET_SUB_SPEC	
		52	50 D0 0007F	MOVL	R0, R2	
		0E	52 D1 00082	CMPL	R2, #14	6985
			0F 12 00085	BNEQ	3\$	
			10 AE 9F 00087	PUSHAB	COLL_DESC	6988
			08 AE 9F 0008A	PUSHAB	SUB_DESC	
	00000000G	00	02 FB 0008D	CALLS	#2, PM\$PARSE_MOD	
			12 11 00094	BRB	4\$	
		0F	52 D1 00096	CMPL	R2, #15	6999
			13 12 00099	BNEQ	5\$	
			10 AE 9F 0009B	PUSHAB	COLL_DESC	7002
			08 AE 9F 0009E	PUSHAB	SUB_DESC	
	00000000G	00	02 FB 000A1	CALLS	#2, PI\$PARSE_IGN	
			50 D5 000AB	TSTL	R0	
			29 12 000AA	BNEQ	9\$	
			30 11 000AC	BRB	11\$	7005
		10	52 D1 000AE	CMPL	R2, #16	7013
			0C 12 000B1	BNEQ	6\$	
			10 AE 9F 000B3	PUSHAB	COLL_DESC	7019

00000000G	00	01	FB	000B6	CALLS	#1, DF\$DO_FOLD	:	7021	
	11	16	11	000BD	BRB	9\$	:	7025	
		52	D1	000BF	6\$:	CMPL	R2, #17		
0080	CB	07	12	000C2	BNEQ	7\$	:	7028	
		01	88	000C4	BISB2	#1, 128(CA)	:	7030	
	12	0A	11	000C9	BRB	9\$	:	7034	
		52	D1	000CB	7\$:	CMPL	R2, #18		
0080	CB	0E	12	000CE	8\$:	BNEQ	11\$		
55		01	8A	000D0	BICB2	#1, 128(CA)	:	7037	
		01	D0	000D5	9\$:	MOVL	#1, MODIFIED	7039	
		86	11	000D8	BRB	2\$	:	7051	
		64	D5	000DA	10\$:	TSTL	(R4)	7066	
		0A	15	000DC	BLEQ	12\$	:		
	001C818C	8F	DD	000DE	11\$:	PUSHL	#1868172	7069	
		53	DD	000E4	PUSHL	KYW_LINE	:		
		57	11	000E6	BRB	16\$	:		
	0E	0080	CB	E9	000E8	12\$:	BLBC	128(CA), 13\$	7073
		01	DD	000ED	PUSHL	#1	:	7081	
	14	AE	9F	000EF	PUSHAB	COLL_DESC	:		
00000000G	00	02	FB	000F2	CALLS	#2, COLL\$TIE_BREAK	:		
		0C	11	000F9	BRB	14\$	:		
		7E	D4	000FB	13\$:	CLRL	-(SP)	7089	
		14	AE	9F	000FD	PUSHAB	COLL_DESC		
00000000G	00	02	FB	00100	CALLS	#2, COLL\$UPPER	:		
	56	50	D0	00107	14\$:	MOVL	R0, S		
	2E	56	E9	0010A	BLBC	S, 15\$	:	7091	
	6E	01	B0	0010D	MOVW	#1, PAD_CHAR	:	7094	
02	AE	0101	CB	90	00110	MOVB	257(CA), PAD_CHAR+2	7096	
		5E	DD	00116	PUSHL	SP	:	7097	
		14	AE	9F	00118	PUSHAB	COLL_DESC		
00000000G	00	02	FB	0011B	CALLS	#2, COLL\$PAD	:		
	56	50	D0	00122	MOVL	R0, S	:		
	13	56	E9	00125	BLBC	S, 15\$	:	7098	
		10	AE	9F	00128	PUSHAB	COLL_DESC	7100	
		14	AE	9F	0012B	PUSHAB	COLL_DESC		
00000000G	00	02	FB	0012E	CALLS	#2, COLL\$RESULT	:		
	56	50	D0	00135	MOVL	R0, S	:		
	0D	56	E8	00138	BLBS	S, 17\$	:	7101	
		56	DD	0013B	15\$:	PUSHL	S		
		67	DD	0013D	PUSHL	(R7)	:		
00000000G	00	02	FB	0013F	16\$:	CALLS	#2, EC\$ERR_CNTL		
		17	11	00146	BRB	18\$	:		
6C	AB	10	AE	D0	00148	17\$:	MOVL	COLL_DESC, 108(CA)	7105
		10	AE	DD	0014D	PUSHL	COLL_DESC	7106	
00000000G	00	01	FB	00150	CALLS	#1, SASSPC_ALLOC	:		
	68	50	D0	00157	MOVL	R0, 104(CA)	:		
	50	01	D0	0015B	MOVL	#1, R0	:	7108	
			04	0015E	RET		:		
		50	D4	0015F	18\$:	CLRL	R0	7109	
			04	00161	RET		:		

; Routine Size: 354 bytes, Routine Base: SOR\$RO_CODE + 0000

: 4120 7110 1
: 4121 7111 1
: 4122 7112 1 ! ELUDOM


```
: 4123      7113 1 !
: 4124      7114 1 END
: 4125      7115 0 ELUDOM
```

PSECT SUMMARY

```
:
:      Name      Bytes      Attributes
:
:  SOR$RO_CODE      354  NOVEC,NOWRT, RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
```

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	103	25	34	00:00.1
-\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	11	9	12	00:00.1
-\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.0

COMMAND QUALIFIERS

```
:
:  BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:SRTSPCBLI/OBJ=OBJ$:SRTSPCBLI MSRC$:SRTSPCBLI/UPDATE=(ENH$:SRTSPCBLI
: )
```

```
: 4126      L 7116 0 MODULE PK$PAR  (IDENT = 'T03-038' %REQUIRE('SRC$:DEFS0') ) =
: 4127      7122 1 BEGIN
: 4128      7123 1 REQUIRE 'SRC$:DKS';
: 4129      7421 1 EXTERNAL ROUTINE
: 4130      7422 1      ERR_CNTL:      CA_LINKAGE,      ! Error control routine
: 4131      7423 1      SPC-ALLOC:    CA_LINKAGE,      ! Allocate storage from the work area
: 4132      7424 1      SKIP_IGNORED,      ! Skip ignored text characters
: 4133      7425 1      SKIP-COMMA,      ! Skip commas and ignored stuff
: 4134      7426 1      GET_RYW_TYPE,      ! Get keyword type
: 4135      7427 1      GET-STRING,      ! Find string and move around it
: 4136      7428 1      SEARCH_TABLE;      ! Search a table for a string
```

```

: 4138      7429 1 ROUTINE PRESUMED_LENGTH( X: REF VECTOR[3], LEN: REF VECTOR[1] ) =
: 4139      7430 BEGIN
: 4140      7431 SELECTONE CH$RCHAR(.X[_PTR]) OF
: 4141      7432 SET
: 4142      7433 [C_QUOTE]: ! String length - 1/2 imbedded quotes
: 4143      7434 BEGIN
: 4144      7435 LOCAL Q;
: 4145      7436 LEN[0] = .X[_LEN] - 2;
: 4146      7437 Q = 0;
: 4147      7438 DECR I FROM .LEN[0] TO 1 DO
: 4148      7439 IF CH$RCHAR(.X[_PTR] + .I) EQL C_QUOTE THEN Q = .Q + 1;
: 4149      7440 LEN[0] = .LEN[0] - .Q ^ -1;
: 4150      7441 END;
: 4151      7442 [C_PERCENT]: LEN[0] = 1;
: 4152      7443 [XC'O' TO XC'9']: LEN[0] = 1;
: 4153      7444 [OTHERWISE]: RETURN FAIL;
: 4154      7445 TES;
: 4155      7446
: 4156      7447 RETURN SUCCESS
: 4157      7448 END;

```

```

.TITLE PK$PAR
.IDENT \T03-038\

.EXTRN EC$ERR CNTL, SASSPC_ALLOC
.EXTRN SI$SKIP_IGNORED
.EXTRN SC$SKIP_COMMA, GKT$GET_KYW_TYPE
.EXTRN GS$GET_STRING, ST$SEARCH_TABLE

.PSECT SOR$RO_CODE,NOWRT, SHR, PIC,2

```

```

                                0004 00000 PRESUMED_LENGTH:
                                .WORD Save R2
                                51      04 AC D0 00002      MOVL X, R1      : 7429
                                52      04 A1 D0 00006      MOVL 4(R1), R2      : 7431
                                50      62 9A 0000A      MOVZBL (R2), R0
                                22      50 91 0000D      CMPB R0, #34
                                24      12 00010      BNEQ 3$      : 7433
                                08 BC      61      02 C3 00012      SUBL3 #2, (R1), @LEN      : 7436
                                50      D4 00017      CLRL Q      : 7437
                                51      08 BC      01 C1 00019      ADDL3 #1, @LEN, I      : 7438
                                22      08 11 0001E      BRB 2$
                                6142      91 00020 1$:      CMPB (1)(R2), #34      : 7439
                                02      12 00024      BNEQ 2$
                                50      D6 00026      INCL Q
                                51      F5 00028 2$:      SOBGTR I, 1$
                                50      FF 8F 78 0002B      ASHL #-1, Q, R0      : 7440
                                08 BC      50 C2 00030      SUBL2 R0, @LEN
                                13      11 00034      BRB 5$
                                25      50 91 00036 3$:      CMPB R0, #37
                                0A      13 00039      BEQL 4$
                                30      50 91 0003B      CMPB R0, #48      : 7442
                                0D      1F 0003E      BLSSU 6$
                                39      50 91 00040      CMPB R0, #57
                                08      1A 00043      BGTRU 6$
                                08 BC      01 D0 00045 4$:      MOVL #1, @LEN      : 7443

```


L⁸
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

Page 138
(32)

```

01  D0 00049 5$:      MOVL      #1, R0
    04 0004C          RET
50  D4 0004D 6$:      CLRL      R0
    04 0004F          RET

```

: 7447
:
:
: 7448
:

```
; Routine Size: 80 bytes,    Routine Base: SOR$RO_CODE + 0000
```

```
: 4159      7449 1 GLOBAL ROUTINE PARSE KEY(  
: 4160      7450 1     KEY: REF VECTOR[3],  
: 4161      7451 1     IS DATA,  
: 4162      7452 1     CON_SYM:REF CON_SYM_TAB[],  
: 4163      7453 1     CON_CNT:REF VECTOR[1],  
: 4164      7454 1     FLD_SYM:REF SYM_TAB[],  
: 4165      7455 1     CDX_SYM:REF SYM_TAB[],  
: 4166      7456 1     CDX_CNT:REF VECTOR[1]): CA_LINKAGE =  
: 4167      7457 1  
: 4168      7458 1 ++  
: 4169      7459 1 Functional Description:  
: 4170      7460 1  
: 4171      7461 1     Parse a key or data clause  
: 4172      7462 1  
: 4173      7463 1 Formal Parameters:  
: 4174      7464 1  
: 4175      7465 1     KEY      - Address of the length/address/line of the key clause.  
: 4176      7466 1     IS DATA - True iff a DATA clause is being parsed  
: 4177      7467 1     CON_SYM  - Address of constant symbol table.  
: 4178      7468 1     CON_CNT  - Address containing number of entries already in table.  
: 4179      7469 1     FLD_SYM  - Address of field symbol table.  
: 4180      7470 1     CDX_SYM  - Address of field symbol table.  
: 4181      7471 1     CDX_CNT  - Address of number of entries in table.  
: 4182      7472 1  
: 4183      7473 1 Implicit Inputs:  
: 4184      7474 1  
: 4185      7475 1 Implicit Outputs:  
: 4186      7476 1  
: 4187      7477 1 Routine Value:  
: 4188      7478 1  
: 4189      7479 1     SUCCESS  
: 4190      7480 1     FAIL  
: 4191      7481 1  
: 4192      7482 1 Side Effects:  
: 4193      7483 1  
: 4194      7484 1 Notes:  
: 4195      7485 1     The syntax parsed by this routine is:  
: 4196      7486 1  
: 4197      7487 1     s=else  
: 4198      7488 1     {  
: 4199      7489 1         [  
: 4200      7490 1             IF cond THEN  
: 4201      7491 1             s=if  
: 4202      7492 1         ]  
: 4203      7493 1         field  
: 4204      7494 1         if s=else then exitloop  
: 4205      7495 1         [  
: 4206      7496 1             ELSE  
: 4207      7497 1             s=else  
: 4208      7498 1         ]  
: 4209      7499 1     }...  
: 4210      7500 1  
: 4211      7501 1 --  
: 4212      7502 2 BEGIN  
: 4213      7503 2  
: 4214      7504 2 LOCAL  
: 4215      7505 2 INDEX,
```



```
: 4216      7506      2      LENGTH,      ! Max length conditional key/data
: 4217      7507      2      SAVE_COUNT,      ! Save the number entries in constant table
: 4218      7508      2      FIRST_IDX,      ! KFT index of first entry for this key
: 4219      7509      2      STATE,      ! State variable
: 4220      7510      2      DESCEND,      ! Direction is descending
: 4221      7511      2      LOC_KEY: VECTOR[3],      ! Local length/address into key clause
: 4222      7512      2      TEMP_KEY: VECTOR[3],      ! Temporary length/address into key clause
: 4223      7513      2      KYW_TYPE;      ! Keyword type
: 4224      7514      2      LOCAL
: 4225      7515      2      KFT: REF KFT_TAB[];
: 4226      7516      2
: 4227      7517      2      CA_AREA_( CA );
: 4228      7518      2
: 4229      7519      2      output_('parse_key');
: 4230      7520      2
: 4231      7521      2      ! Remember the starting count
: 4232      7522      2      SAVE_COUNT = .CON_CNT[0];
: 4233      7523      2      LENGTH = 0;
: 4234      7524      2
: 4235      7525      2      ! Default to ascending (not descending)
: 4236      7526      2      DESCEND = FALSE;
: 4237      7527      2
: 4238      7528      2      ! Set the index of the first KFT entry we will allocate
: 4239      7529      2      FIRST_IDX = .CA[CA_KFT_SIZ];
: 4240      7530      2
: 4241      7531      2      ! Act (pretty much) like we've just seen an ELSE
: 4242      7532      2      !
: 4243      7533      2      STATE = KW_ELSE;
: 4244      7534      2
: 4245      7535      2      ! Get what may be a field name
: 4246      7536      2      IF SKIP_IGNORED( KEY[0] ) EQL FAIL THEN RETURN FAIL;
: 4247      7537      2      IF GET_STRING( KEY[0], LOC_KEY[0] ) EQL FAIL THEN RETURN FAIL;
: 4248      7538      2
: 4249      7539      2      ! While there are more key sub-clauses
: 4250      7540      2      WHILE TRUE DO
: 4251      7541      2      BEGIN
: 4252      7542      2
: 4253      7543      2      ! Allocate a KFT entry
: 4254      7544      2      KFT = ALLOC (KFT_UNIT);
: 4255      7545      2      IF .CA[CA_KFT_SIZ] GEQ KFT_MAX THEN ERR_(0, SORS_WKAREA);
: 4256      7546      2      CA[CA_KFT_SIZ] = .CA[CA_KFT_SIZ] + 1;
: 4257      7547      2
: 4258      7548      2      ! Initialize the key/data field table entry
: 4259      7549      2      !
: 4260      7550      2      KFT[0, KFT_DATA] = .IS_DATA;
: 4261      7551      2      KFT[0, KFT_BUILD] = TRUE;
: 4262      7552      2      KFT[0, KFT_CONTINUE] = TRUE;
: 4263      7553      2      KFT[0, KFT_CONT_CDX] = TRUE;
: 4264      7554      2      KFT[0, KFT_NDE_POS] = 0;      ! needed?
: 4265      7555      2      KFT[0, KFT_FDT_IDX] = 0;      ! needed?
: 4266      7556      2      KFT[0, KFT_TDT_IDX] = 0;      ! needed?
: 4267      7557      2      KFT[0, KFT_CONSTANT] = FALSE;
: 4268      7558      2      KFT[0, KFT_CONDX] = FALSE;
: 4269      7559      2      KFT[0, KFT_DESCEND] = FALSE;
: 4270      7560      2
: 4271      7561      2      ! Try scanning "IF cond THEN"
: 4272      7562      2      !
```



```
: 4273      7563      3      TEMP_KEY[ PTR] = .LOC_KEY[ PTR];
: 4274      7564      3      TEMP_KEY[ LEN] = .LOC_KEY[ LEN];
: 4275      7565      3      TEMP_KEY[ LINE] = .LOC_KEY[ LINE];
: 4276      7566      3      KYW_TYPE = GET_KYW_TYPE( TEMP_KEY[0] );
: 4277      7567      3      IF .KYW_TYPE EQL KW_IF
: 4278      7568      3      THEN
: 4279      7569      4          BEGIN
: 4280      7570      4              ! This is a conditional key
: 4281      7571      4              KFT[0, KFT_CONDX] = TRUE;
: 4282      7572      4
: 4283      7573      4              ! Get name of condition
: 4284      7574      4              IF SKIP_IGNORED( KEY[0] ) EQL FAIL THEN RETURN FAIL;
: 4285      7575      4              IF GET_STRING( KEY[0], LOC_KEY[0] ) EQL FAIL THEN RETURN FAIL;
: 4286      7576      4
: 4287      7577      4              ! Search condition symbol table for name
: 4288      7578      4              IF SEARCH_TABLE( LOC_KEY[0],
: 4289      7579      4                  INDEX, CDX_SYM[0,BASE_], .CDX_CNT[0] ) EQL FAIL
: 4290      7580      4              THEN
: 4291      7581      4                  RETURN FAIL;
: 4292      7582      4
: 4293      7583      4              ! Put the TDT index in the key field table
: 4294      7584      4              KFT[0, KFT_TDT_IDX] = .CDX_SYM[.INDEX, SYM_INDEX];
: 4295      7585      4
: 4296      7586      4              ! Get the keyword type
: 4297      7587      4              IF SKIP_IGNORED( KEY[0] ) EQL FAIL THEN RETURN FAIL;
: 4298      7588      4              KYW_TYPE = GET_KYW_TYPE( KEY[0] );
: 4299      7589      4
: 4300      7590      4              ! If keyword is not "then"
: 4301      7591      4              IF .KYW_TYPE NEQ KW_THEN THEN RETURN FAIL;
: 4302      7592      4
: 4303      7593      4              ! Get the field name or constant
: 4304      7594      4              IF SKIP_IGNORED( KEY[0] ) EQL FAIL THEN RETURN FAIL;
: 4305      7595      4              IF GET_STRING( KEY[0], LOC_KEY[0] ) EQL FAIL THEN RETURN FAIL;
: 4306      7596      4
: 4307      7597      4              STATE = KW_IF;
: 4308      7598      4              END;
: 4309      7599      3
: 4310      7600      3      ! Search the field symbol table for field name
: 4311      7601      3      IF SEARCH_TABLE( LOC_KEY[0],
: 4312      7602      3          INDEX, FLD_SYM[0,BASE_], .CA[CA_FDT_SIZE] ) EQL SUCCESS
: 4313      7603      3      THEN
: 4314      7604      3          BEGIN
: 4315      7605      3              ! Put the index in the key field table
: 4316      7606      4              KFT[0, KFT_FDT_IDX] = .FLD_SYM[.INDEX, SYM_INDEX];
: 4317      7607      4              END
: 4318      7608      4      ELSE
: 4319      7609      4          BEGIN
: 4320      7610      4              ! It had better be a constant
: 4321      7611      4              ! Put the constant symbol table index in the FDT index
: 4322      7612      3              ! Field of the key field table, this will be the
: 4323      7613      4              ! Same as index into constant field table
: 4324      7614      4
: 4325      7615      4
: 4326      7616      4
: 4327      7617      4
: 4328      7618      4
: 4329      7619      4
```



```
: 4330      7620      4      KFT[0, KFT_FDT_IDX] = .CON_CNT[0];
: 4331      7621      4      KFT[0, KFT_CONSTANT] = TRUE;
: 4332      7622      4
: 4333      7623      4      ! Make entry in constant table
: 4334      7624      4      CON_SYM[.CON_CNT[0], CON_NAM_ADR] = .LOC_KEY[PTR];
: 4335      7625      4      CON_SYM[.CON_CNT[0], CON_NAM_LEN] = .LOC_KEY[_LEN];
: 4336      7626      4      CON_CNT[0] = .CON_CNT[0] + 1;
: 4337      7627      4
: 4338      7628      4      ! Determine the presumed length of the constant.
: 4339      7629      4      ! Note that the actual length depends on other key/data fields.
: 4340      7630      4      ! String constants:      Length of the string
: 4341      7631      4      ! Radix constants:      1
: 4342      7632      4      ! Decimal constants:      1
: 4343      7633      4
: 4344      7634      5      BEGIN
: 4345      7635      5      LOCAL L;
: 4346      7636      5      IF PRESUMED_LENGTH( LOC_KEY[0], L ) EQL FAIL THEN RETURN FAIL;
: 4347      7637      5      LENGTH = MAXU( .LENGTH, .L );
: 4348      7638      4      END;
: 4349      7639      4
: 4350      7640      3      END;
: 4351      7641      3
: 4352      7642      3      ! If we were working on the final "ELSE" clause, then exit the loop
: 4353      7643      3      !
: 4354      7644      3      IF .STATE EQL KW_ELSE THEN EXITLOOP;
: 4355      7645      3
: 4356      7646      3      ! Get the next word (ELSE or IF or nothing)
: 4357      7647      3      !
: 4358      7648      3      SKIP_IGNORED( KEY[0] );
: 4359      7649      3      GET_STRING( KEY[0], LOC_KEY[0] );
: 4360      7650      3
: 4361      7651      3      ! Try scanning "ELSE"
: 4362      7652      3      !
: 4363      7653      3      TEMP_KEY[PTR] = .LOC_KEY[PTR];
: 4364      7654      3      TEMP_KEY[_LEN] = .LOC_KEY[_LEN];
: 4365      7655      3      TEMP_KEY[_LINE] = .LOC_KEY[_LINE];
: 4366      7656      3      KYW_TYPE = GET_KYW_TYPE( TEMP_KEY[0] );
: 4367      7657      3      IF .KYW_TYPE EQL KW_ELSE
: 4368      7658      3      THEN
: 4369      7659      4      BEGIN
: 4370      7660      4      !
: 4371      7661      4      ! Get the next word
: 4372      7662      4      !
: 4373      7663      4      STATE = KW_ELSE;
: 4374      7664      4      IF SKIP_IGNORED( KEY[0] ) EQL FAIL THEN RETURN FAIL;
: 4375      7665      4      IF GET_STRING( KEY[0], LOC_KEY[0] ) EQL FAIL THEN RETURN FAIL;
: 4376      7666      4      END;
: 4377      7667      3
: 4378      7668      2      END;
: 4379      7669      2
: 4380      7670      2      ! Check for a comma, which must be followed by "ASCENDING" or "DESCENDING"
: 4381      7671      2      !
: 4382      7672      2      IF SKIP_COMMA( KEY[0] ) NEQ FAIL
: 4383      7673      2      THEN
: 4384      7674      3      BEGIN
: 4385      7675      3      IF GET_STRING( KEY[0], LOC_KEY[0] ) EQL FAIL THEN RETURN FAIL;
: 4386      7676      3      SELECTONE GET_KYW_TYPE( LOC_KEY[0] ) OF
```

4387 7677 3
4388 7678 3
4389 7679 3
4390 7680 3
4391 7681 3
4392 7682 2
4393 7683 2
4394 7684 2
4395 7685 2
4396 7686 2
4397 7687 2
4398 7688 2
4399 7689 2
4400 7690 2
4401 7691 2
4402 7692 2
4403 7693 2
4404 7694 2
4405 7695 2
4406 7696 3
4407 7697 3
4408 7698 3
4409 7699 2
4410 7700 2
4411 7701 2
4412 7702 2
4413 7703 2
4414 7704 2
4415 7705 3
4416 7706 3
4417 7707 3
4418 7708 2
4419 7709 2
4420 7710 2
4421 7711 2
4422 7712 1

```
SET
[KW_ASCE]: DESCEND = FALSE;
[KW_DESC]: DESCEND = TRUE;
[OTHERWISE]: RETURN FAIL;
TES;
END;
! There had better not be anything else
IF SKIP_IGNORED( KEY[0] ) NEQ FAIL THEN RETURN FAIL;
! Set the last entry's continued condition bit off
KFT[0, KFT_CONT_CDX] = FALSE;
! Set the result length for all entries made in constant table
! On this call. This will be used for creating constant field table
! ??? This does not correctly account for mixing fields and constants.
INCR IN FROM .SAVE_COUNT TO .CON_CNT[0]-1 DO
  BEGIN
    CON_SYMC[IN, CON_LENGTH] = .LENGTH;
    CON_SYMC[IN, CON_INDEX] = 0;
  END;
! Propagate the descending bit
KFT = .CA[CA_KFT_ADR];
KFT = KFT[.FIRST_IDX, BASE_];
DECR IN FROM .CA[CA_KFT_SIZ]-.FIRST_IDX-1 TO 0 DO
  BEGIN
    KFT[0, KFT_DESCEND] = .DESCEND;
    KFT = KFT[T, BASE_];
  END;
RETURN SUCCESS
END;
```

07FC 00000				.ENTRY	PK\$PARSE_KEY, Save R2,R3,R4,R5,R6,R7,R8,R9,-;	
5E		20	C2 00002	SUBL2	R10	7449
53	10	BC	D0 00005	MOVL	#32, SP	
		52	D4 00009	MOVL	@CON_CNT, SAVE_COUNT	7522
		5A	D4 0000B	CLRL	LENGTH	7523
58	00FD	CB	9A 0000D	CLRL	DESCEND	7526
57		2F	D0 00012	MOVZBL	253(CA), FIRST_IDX	7529
55	04	AC	D0 00015	MOVL	#47, STATE	7533
		0185	31 00019	MOVL	KEY, R5	7536
		08	DD 0001C	BRW	11\$	
00000000G	00	01	FB 0001E	PUSHL	#8	7544
	54	50	D0 00025	CALLS	#1, SA\$SPC_ALLOC	
FF	8F	00FD	CB 91 00028	MOVL	R0, KFT	
		17	1F 0002E	CMPB	253(CA), #255	7545
		001C815C	8F DD 00030	BLSSU	3\$	
				PUSHL	#1868124	

			7E	D4	00036	CLRL	-(SP)	:	
			02	FB	00038	CALLS	#2, EC\$ERR_CNTL	:	
			50	D1	0003F	CMPL	R0, #1	:	
			03	13	00042	BEQL	3\$	:	
			0200	31	00044	BRW	21\$	:	
			00FD	CB	96	INCB	253(CA)	:	7546
			03	A4	9E	MOVAB	3(KFT), R6	:	7550
66	01	56	08	AC	F0	INSV	IS DATA, #6, #1, (R6)	:	
		06		15	88	BISB2	#2T, (R6)	:	7553
		66		64	B4	CLRW	(KFT)	:	7554
			04	A4	B4	CLRW	4(KFT)	:	7555
		66		2A	8A	BICB2	#42, (R6)	:	7559
		08	14	AE	7D	MOVQ	LOC_KEY, TEMP_KEY	:	7564
		10	1C	AE	D0	MOVL	LOC_KEY+8, TEMP_KEY+8	:	7565
			08	AE	9F	PUSHAB	TEMP_KEY	:	7566
		00000000G	00	01	FB	CALLS	#1, GKT\$GET_KYW_TYPE	:	
		59		50	D0	MOVL	R0, KYW_TYPE	:	
		2D		59	D1	CMPL	KYW_TYPE, #45	:	7567
			03	13	0007A	BEQL	4\$	:	
			0085	31	0007C	BRW	7\$	:	
		66		08	88	BISB2	#8, (R6)	:	7572
				55	DD	PUSHL	R5	:	7575
		00000000G	00	01	FB	CALLS	#1, SI\$SKIP_IGNORED	:	
				50	D5	TSTL	R0	:	
				B5	13	BEQL	2\$	:	
			14	AE	9F	PUSHAB	LOC_KEY	:	7576
				55	DD	PUSHL	R5	:	
		00000000G	00	02	FB	CALLS	#2, GS\$GET_STRING	:	
				50	D5	TSTL	R0	:	
				A5	13	BEQL	2\$	:	
			1C	BC	DD	PUSHL	@CDX_CNT	:	7580
			18	AC	DD	PUSHL	CDX_SYM	:	
			08	AE	9F	PUSHAB	INDEX	:	7579
			20	AE	9F	PUSHAB	LOC_KEY	:	
		00000000G	00	04	FB	CALLS	#4, ST\$SEARCH_TABLE	:	7580
				50	D5	TSTL	R0	:	
				8E	13	BEQL	2\$	:	
		50		06	C5	MULL3	#6, INDEX, R0	:	7585
		6E		AC	C0	ADDL2	CDX_SYM, R0	:	
		50		A0	90	MOVB	5(R0), 5(KFT)	:	
		05		55	DD	PUSHL	R5	:	7588
		00000000G	00	01	FB	CALLS	#1, SI\$SKIP_IGNORED	:	
				50	D5	TSTL	R0	:	
				11	13	BEQL	5\$	:	
				55	DD	PUSHL	R5	:	7589
		00000000G	00	01	FB	CALLS	#1, GKT\$GET_KYW_TYPE	:	
		59		50	D0	MOVL	R0, KYW_TYPE	:	
		2E		59	D1	CMPL	KYW_TYPE, #46	:	7592
			03	13	000DF	BEQL	6\$	:	
			0163	31	000E1	BRW	21\$	:	
				55	DD	PUSHL	R5	:	7595
		00000000G	00	01	FB	CALLS	#1, SI\$SKIP_IGNORED	:	
				50	D5	TSTL	R0	:	
				F0	13	BEQL	5\$	:	
			14	AE	9F	PUSHAB	LOC_KEY	:	7596
				55	DD	PUSHL	R5	:	
		00000000G	00	02	FB	CALLS	#2, GS\$GET_STRING	:	

		50	D5	000FD	TSTL	R0		
		E0	13	000FF	BEQL	5\$		
	57	2D	D0	00101	MOVL	#45, STATE		7598
	7E	CB	9A	00104	MOVZBL	255(CA), -(SP)		7604
		14	AC	DD 00109	PUSHL	FLD SYM		
		08	AE	9F 0010C	PUSHAB	INDEX		7603
		20	AE	9F 0010F	PUSHAB	LOC_KEY		
00000000G	00	04	FB	00112	CALLS	#4, ST\$SEARCH_TABLE		7604
	01	50	D1	00119	CMPL	R0, #1		
50	6E	0F	12	0011C	BNEQ	8\$		
	50	06	C5	0011E	MULL3	#6, INDEX, R0		7610
	04	AC	C0	00122	ADDL2	FLD SYM, R0		
		05	A0	90 00126	MOVB	5(R0), 4(KFT)		
			3B	11 0012B	BRB	10\$		7603
	50	10	BC	D0 0012D	MOVL	@CON_CNT, R0		7620
	04		50	90 00131	MOVB	R0, 4(KFT)		
			02	88 00135	BISB2	#2, (R6)		7621
	66		7E	00138	MOVAQ	@CON_SYM[R0], R0		7624
	50	0C	BC	40 0013D	MOVL	LOC_KEY+4, (R0)		
	60	18	AE	D0 00141	MOVB	LOC_KEY, 4(R0)		7625
	04	14	AE	90 00146	INCL	@CON_CNT		7626
		10	BC	D6 00149	PUSHAB	L		7636
		04	AE	9F 0014C	PUSHAB	LOC_KEY		
		18	AE	9F 0014F	CALLS	#2, PRESUMED_LENGTH		
FE5C	CF		02	FB 00154	TSTL	R0		
			50	D5 00156	BEQL	13\$		
	50		52	D0 00158	MOVL	LENGTH, R0		7637
	04		50	D1 0015B	CMPL	R0, L		
			04	1E 0015F	BGEQU	9\$		
	50	04	AE	D0 00161	MOVL	L, R0		
	52		50	D0 00165	MOVL	R0, LENGTH		
	2F		57	D1 00168	CMPL	STATE, #47		7644
			57	13 0016B	BEQL	14\$		
00000000G	00		55	DD 0016D	PUSHL	R5		7648
			01	FB 0016F	CALLS	#1, SI\$SKIP_IGNORED		
		14	AE	9F 00176	PUSHAB	LOC_KEY		7649
			55	DD 00179	PUSHL	R5		
00000000G	00		02	FB 0017B	CALLS	#2, GS\$GET_STRING		
	08	14	AE	7D 00182	MOVQ	LOC_KEY, TEMP_KEY		7654
	10	1C	AE	D0 00187	MOVL	LOC_KEY+8, TEMP_KEY+8		7655
		08	AE	9F 0018C	PUSHAB	TEMP_KEY		7656
00000000G	00		01	FB 0018F	CALLS	#1, GKT\$GET_KYW_TYPE		
	59		50	D0 00196	MOVL	R0, KYW_TYPE		
	2F		59	D1 00199	CMPL	KYW_TYPE, #47		7657
			1E	12 0019C	BNEQ	12\$		
	57		2F	D0 0019E	MOVL	#47, STATE		7663
			55	DD 001A1	PUSHL	R5		7664
00000000G	00		01	FB 001A3	CALLS	#1, SI\$SKIP_IGNORED		
			50	D5 001AA	TSTL	R0		
			13	13 001AC	BEQL	13\$		
		14	AE	9F 001AE	PUSHAB	LOC_KEY		7665
			55	DD 001B1	PUSHL	R5		
00000000G	00		02	FB 001B3	CALLS	#2, GS\$GET_STRING		
			50	D5 001BA	TSTL	R0		
			03	13 001BC	BEQL	13\$		
		FE5B	31	001BE	BRW	1\$		
		0083	31	001C1	BRW	21\$		

00000000G	00	55	DD	001C4	14\$:	PUSHL	R5		7672
		01	FB	001C6		CALLS	#1, SC\$SKIP_COMMA		
		50	D5	001CD		TSTL	R0		
		2B	13	001CF		BEQL	16\$		
		14	AE	9F	001D1	PUSHAB	LOC_KEY		7675
00000000G	00	55	DD	001D4		PUSHL	R5		
		02	FB	001D6		CALLS	#2, GS\$GET_STRING		
		50	D5	001DD		TSTL	R0		
		66	13	001DF		BEQL	21\$		
00000000G	00	14	AE	9F	001E1	PUSHAB	LOC_KEY		7676
	30	01	FB	001E4		CALLS	#1, GKT\$GET_KYW_TYPE		
		50	D1	001EB		CMPL	R0, #48		7678
		04	12	001EE		BNEQ	15\$		
		5A	D4	001F0		CLRL	DESCEND		
	31	08	11	001F2		BRB	16\$		
		50	D1	001F4	15\$:	CMPL	R0, #49		7679
	5A	4E	12	001F7		BNEQ	21\$		
		01	D0	001F9		MOVL	#1, DESCEND		
00000000G	00	55	DD	001FC	16\$:	PUSHL	R5		7686
		01	FB	001FE		CALLS	#1, SI\$SKIP_IGNORED		
		50	D5	00205		TSTL	R0		
		3E	12	00207		BNEQ	21\$		
03	A4	04	8A	00209		BICB2	#4, 3(KFT)		7689
	50	FF	A3	9E	0020D	MOVAB	-1(R3), IN		7697
		0C	11	00211		BRB	18\$		
	51	0C	BC	40	7E	00213	17\$:		
06	A1	52	90	00218		MOVAQ	@CON_SYM[IN], R1		
		05	A1	94	0021C	MOVB	LENGTH, 6(R1)		
		10	BC	F2	0021F	18\$:	AOBLSS	@CON_CNT, IN, 17\$	7698
EF	50	0108	CB	D0	00224	18\$:	MOVL	264(CA), KFT	7695
	54	64	48	7E	00229	MOVAQ	(KFT)[FIRST_IDX], KFT		7702
	54	00	FD	CB	9A	0022D	MOVZBL	253(CA), R0	7703
	50	58	C2	00232		SUBL2	FIRST_IDX, R0		7704
	50	09	11	00235		BRB	20\$		
03	A4	01	05	5A	F0	00237	19\$:		7706
			54	08	C0	0023D	INSV	DESCEND, #5, #1, 3(KFT)	
			F4	50	F4	00240	20\$:		7707
			50	01	D0	00243	20\$:		7704
					04	00246	MOV	#1, R0	7710
					50	D4	00247	21\$:	
					04	00249	CLRL	R0	7712
							RET		

; Routine Size: 586 bytes, Routine Base: SOR\$R0_CODE + 0050

: 4423	7713	1	
: 4424	7714	1	
: 4425	7715	1	! ELUDOM
: 4426	7716	1	! ELUDOM
: 4427	7717	1	END
: 4428	7718	0	ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
SORSRO_CODE	666	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	103	25	34	00:00.1
\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	24	20	12	00:00.1
\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LISS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI)

```

: 4429      L 7719 0 MODULE PT$PAR  (IDENT = 'T03-038' %REQUIRE('SRC$:DEFS0')) =
: 4430      7725 1 BEGIN
: 4431      7726 1 REQUIRE 'SRC$:DKS';
: 4432      8024 1 EXTERNAL ROUTINE
: 4433      8025 1      ERR_CNTL:      CA_LINKAGE,      ! Error control routine
: 4434      8026 1      SPC_ALLOC:    CA_LINKAGE,      ! Allocate storage from the work area
: 4435      8027 1      SKIP_IGNORED, ! Skip ignored text characters
: 4436      8028 1      GET_RYW TYPE,  ! Get keyword type
: 4437      8029 1      GET_STRING,    ! Find string and move around it
: 4438      8030 1      SEARCH_TABLE; ! Search a table for a string

```



```
4440 8031 1 GLOBAL ROUTINE PARSE TEST(  
4441 8032 1     TST: REF VECTOR[3],  
4442 8033 1     CON_SYM: REF CON_SYM_TAB[],  
4443 8034 1     CON_CNT: REF VECTOR[1],  
4444 8035 1     FLD_SYM: REF SYM_TAB[] ): CA_LINKAGE =  
4445 8036 1  
4446 8037 1 ++  
4447 8038 1 Functional Description:  
4448 8039 1  
4449 8040 1     Parse a test clause  
4450 8041 1  
4451 8042 1 Formal Parameters:  
4452 8043 1  
4453 8044 1     TST      - Address of length/address/line of the test clause.  
4454 8045 1     CON_SYM  - Address of constant symbol table.  
4455 8046 1     CON_CNT  - Address containing number of entries already in table.  
4456 8047 1     FLD_SYM  - Address of field symbol table.  
4457 8048 1  
4458 8049 1 Implicit Inputs:  
4459 8050 1  
4460 8051 1 Implicit Outputs:  
4461 8052 1  
4462 8053 1 Routine Value:  
4463 8054 1  
4464 8055 1     SUCCESS  
4465 8056 1     FAIL  
4466 8057 1  
4467 8058 1 Side Effects:  
4468 8059 1  
4469 8060 1 Notes:  
4470 8061 1  
4471 8062 1     A "global" area is initialized so that the recursive subroutines can  
4472 8063 1     access the various parameters they need.  
4473 8064 1     The GET_xxx routines parse the condition and produce TDT table entries,  
4474 8065 1     and a postfix representation of the condition (in S2).  
4475 8066 1     Finally, CEVAL is called to process the postfix expression (from right  
4476 8067 1     to left), adjusting the TDT GOTOs and (possibly) negating the tests.  
4477 8068 1     See the comments in SRTSPC.REQ that describe how the TDT table is used.  
4478 8069 1 --  
4479 8070 2 BEGIN  
4480 8071 2  
4481 8072 2 MACRO  
4482 8073 2     LB_DECL = EXTERNAL REGISTER LB: REF VECTOR %;  
4483 8074 2  
4484 8075 2 FORWARD ROUTINE  
4485 8076 2     GET_EXPR:      CA_LINKAGE_LB, ! Parse an expression  
4486 8077 2     GET_SIMPLE:     CA_LINKAGE_LB;  ! Parse a simple condition  
4487 8078 2  
4488 8079 2 MACRO  
4489 8080 2     STACK2(X) =  
4490 8081 2         BEGIN  
4491 8082 2             S2[S2I] = X;  
4492 8083 2             S2I = S2I + 1;  
4493 8084 2             IF S2I GEQ S2_MAX THEN RETURN FAIL;  
4494 8085 2             END %  
4495 8086 2     S1I = CA[CA_TDT_S1Z] %,  
4496 8087 2     S2(X) = (LB[K_S2]+X)<0,8,0> %.
```

4497 8088 2
4498 8089 2
4499 8090 2
4500 8091 2
4501 8092 2
4502 8093 2
4503 8094 2
4504 8095 2
4505 8096 2
4506 8097 2
4507 8098 2
4508 8099 2
4509 8100 2
4510 8101 2
4511 8102 2
4512 8103 2
4513 8104 2
4514 8105 2
4515 8106 2
4516 8107 2
4517 8108 2
4518 8109 2
4519 8110 2
4520 8111 2
4521 8112 2
4522 8113 2
4523 8114 2
4524 8115 2
4525 8116 2
4526 8117 2
4527 8118 2
4528 8119 2
4529 8120 2
4530 8121 2
4531 8122 2
4532 8123 2
4533 8124 2
4534 8125 2

```
S2I = LB[K_S2I] %;

LITERAL
S2_MAX = 30,      ! Bound on max complexity of a test
OP_AND = 0,       ! AND operator
OP_OR  = 1,       ! OR operator
OP_NEG = 2,       ! NOT operator
OP_SYM = 3,       ! Simple condition
K_TST  = 0,
K_S2I  = 1,
K_CON_SYM = 2,
K_CON_CNT = 3,
K_FLD_SYM = 4,
K_S2    = 5;

ROUTINE GET_KYW_TYPE_NM( KW ): CA_LINKAGE_LB =
BEGIN
  LOCAL
    TST: REF VECTOR[3],
    SKEY: VECTOR[3];
  LB_DECL;
  CA_AREA( CA );
  TST = .LB[K_TST];
  SKEY[0] = .TST[0];
  SKEY[1] = .TST[1];
  SKEY[2] = .TST[2];
  IF SKIP_IGNORED( TST[0] ) EQL FAIL THEN RETURN FAIL;
  IF GET_KYW_TYPE( TST[0] ) NEQ .KW
  THEN
    BEGIN
      TST[0] = .SKEY[0];
      TST[1] = .SKEY[1];
      TST[2] = .SKEY[2];
      RETURN FAIL;
    END
  ELSE
    RETURN SUCCESS;
  END;
END;
```

.TITLE PT\$PAR
.IDENT \T03-038\

.EXTRN EC\$ERR_CNTL, SA\$SPC_ALLOC
.EXTRN SI\$SKIP_IGNORED
.EXTRN GKT\$GET_KYW_TYPE
.EXTRN GS\$GET_STRING, ST\$SEARCH_TABLE

.PSECT SOR\$RO_CODE, NOWRT, SHR, PIC, 2

0004 00000 GET_KYW_TYPE_NM:

5E	08	C2	00002	.WORD	Save R2	8103
52	64	D0	00005	SUBL2	#8, SP	8110
	62	DD	00008	MOVL	(LB), TST	8111
04	A2	7D	0000A	PUSHL	(TST)	8112
	52	DD	0000F	MOVQ	4(TST), SKEY+4	8114
				PUSHL	TST	

00000000G	00	01	FB	00011	CALLS	#1, SI\$SKIP_IGNORED	:
		50	D5	00018	TSTL	R0	:
		1D	13	0001A	BEQL	2\$	:
		52	DD	0001C	PUSHL	TST	8115
00000000G	00	01	FB	0001E	CALLS	#1, GKT\$GET_KYW_TYPE	:
04	AC	50	D1	00025	CML	R0, KW	:
		0A	13	00029	BEQL	1\$	:
	62	6E	7D	0002B	MOVQ	SKEY, (TST)	8118
08	A2	AE	D0	0002E	MOVL	SKEY+8, 8(TST)	8120
		04	11	00033	BRB	2\$	8124
	50	01	D0	00035	MOVL	#1, R0	:
			04	00038	RET		:
		50	D4	00039	CLRL	R0	8125
			04	0003B	RET		:

; Routine Size: 60 bytes, Routine Base: SOR\$RO_CODE + 0000

: 4535	8126	2	ROUTINE GET_FACT: CA_LINKAGE_LB =
: 4536	8127	2	BEGIN
: 4537	8128	3	LOCAL
: 4538	8129	3	TST: REF VECTOR[3];
: 4539	8130	3	LB_DECL;
: 4540	8131	3	CA_AREA (CA);
: 4541	8132	3	TST = .[BCK TST];
: 4542	8133	3	IF SKIP_IGNORED(TST[0]) EQL FAIL THEN RETURN FAIL;
: 4543	8134	3	IF CH\$RCHAR(.TST[PTR]) EQL C_L_PAREN
: 4544	8135	3	THEN
: 4545	8136	3	BEGIN
: 4546	8137	4	TST[LEN] = .TST[LEN] - 1;
: 4547	8138	4	TST[PTR] = .TST[PTR] + 1;
: 4548	8139	4	IF NOT GET_EXPR() THEN RETURN FAIL;
: 4549	8140	4	IF SKIP_IGNORED(TST[0]) EQL FAIL THEN RETURN FAIL;
: 4550	8141	4	IF CH\$RCHAR(.TST[PTR]) NEQ C_R_PAREN THEN RETURN FAIL;
: 4551	8142	4	TST[LEN] = .TST[LEN] - 1;
: 4552	8143	4	TST[PTR] = .TST[PTR] + 1;
: 4553	8144	4	END
: 4554	8145	4	%IF %DECLARED(KW NOT) %THEN
: 4555	8146	4	ELSE IF GET_KYW_TYPE_NM(KW_NOT);
: 4556	8147	4	THEN
: 4557	8148	4	BEGIN
: 4558	8149	4	IF NOT GET_FACT() THEN RETURN FAIL;
: 4559	8150	4	STACK2(OP_NOT);
: 4560	8151	4	END
: 4561	8152	4	%FI
: 4562	8153	4	ELSE
: 4563	8154	3	BEGIN
: 4564	8155	4	IF NOT GET_SIMPLE() THEN RETURN FAIL;
: 4565	8156	4	STACK2(OP_SYM);
: 4566	8157	4	END;
: 4567	8158	3	RETURN SUCCESS;
: 4568	8159	3	END;
: 4569	8160	2	

```
000C 00000 GET_FACT:
53 00000000G 00 9E 00002 .WORD Save R2,R3      : 8127
52          64 D0 00009 MOVAB SI$SKIP_IGNORED, R3 : 8133
          52 DD 0000C MOVL (LB), TST      : 8134
63          01 FB 0000E CALLS #1, SI$SKIP_IGNORED
          50 D5 00011 TSTL R0
          47 13 00013 BEQL 3$
28          04 B2 91 00015 CMPB @4(TST), #40      : 8135
          23 12 00019 BNEQ 1$
          62 D7 0001B DECL (TST)                  : 8138
          04 A2 D6 0001D INCL 4(TST)                : 8139
0000V CF      00 FB 00020 CALLS #0, GET_EXPR        : 8140
34          50 E9 00025 BLBC R0, 3$
          52 DD 00028 PUSHL TST
63          01 FB 0002A CALLS #1, SI$SKIP_IGNORED
          50 D5 0002D TSTL R0
          28 13 0002F BEQL 3$
29          04 B2 91 00031 CMPB @4(TST), #41      : 8142
          25 12 00035 BNEQ 3$
          62 D7 00037 DECL (TST)                  : 8143
          04 A2 D6 00039 INCL 4(TST)                : 8144
          1A 11 0003C BRB 2$
0000V CF      00 FB 0003E 1$: CALLS #0, GET_SIMPLE
16          50 E9 00043 BLBC R0, 3$
50          54 04 A4 C1 00046 ADDL3 4(LB), LB, R0 : 8157
14 A0          03 90 0004B MOVB #3, 20(R0)
          04 A4 D6 0004F INCL 4(LB)
          1E 04 A4 D1 00052 CMPL 4(LB), #30
          04 18 00056 BGEQ 3$
          50 01 D0 00058 2$: MOVL #1, R0
          04 0005B RET
          50 D4 0005C 3$: CLRL R0
          04 0005E RET
```

; Routine Size: 95 bytes, Routine Base: SOR\$RO_CODE + 003C

```
: 4570      8161 2
: 4571      8162 2
: 4572      8163 3
: 4573      8164 3
: 4574      8165 3
: 4575      8166 3
: 4576      8167 3
: 4577      8168 4
: 4578      8169 4
: 4579      8170 4
: 4580      8171 3
: 4581      8172 3
: 4582      8173 2

ROUTINE GET_TERM: CA_LINKAGE_LB =
BEGIN
LB_DECL;
CA_AREA ( CA );
IF NOT GET_FACT() THEN RETURN FAIL;
WHILE GET_RYW_TYPE_NM( KW_AND ) DO
BEGIN
IF NOT GET_FACT() THEN RETURN FAIL;
STACK2(OP_AND);
END;
RETURN SUCCESS;
END;
```


0000 00000 GET_TERM:				WORD	Save nothing	
9B	AF	00	FB 00002	CALLS	#0, GET_FACT	8162
	28	50	E9 00006	BLBC	R0, 3\$	8166
		3B	DD 00009 1\$:	PUSHL	#5\$	8167
FF55	CF	01	FB 0000B	CALLS	#1, GET_KYW_TYPE_NM	
	1A	50	E9 00010	BLBC	R0, 2\$	
8A	AF	00	FB 00013	CALLS	#0, GET_FACT	8169
	17	50	E9 00017	BLBC	R0, 3\$	
50	54	04	A4 C1 0001A	ADDL3	4(LB), LB, R0	8170
		14	A0 94 0001F	CLRB	20(R0)	
		04	A4 D6 00022	INCL	4(LB)	
	1E	04	A4 D1 00025	CMPL	4(LB), #30	
		DE	19 00029	BLSS	1\$	
		04	11 0002B	BRB	3\$	
	50	01	D0 0002D 2\$:	MOVL	#1, R0	8172
			04 00030	RET		
		50	D4 00031 3\$:	CLRL	R0	8173
			04 00033	RET		

; Routine Size: 52 bytes, Routine Base: SOR\$R0_CODE + 009B

: 4583	8174	2	ROUTINE GET_EXPR: CA_LINKAGE_LB = BEGIN LB_DECL; CA_AREA (CA); IF NOT GET_TERM() THEN RETURN FAIL; WHILE GET_KYW_TYPE_NM(KW_OR) DO BEGIN IF NOT GET_TERM() THEN RETURN FAIL; STACK2(OP_OR); END; RETURN SUCCESS; END;
: 4584	8175	2	
: 4585	8176	3	
: 4586	8177	3	
: 4587	8178	3	
: 4588	8179	3	
: 4589	8180	3	
: 4590	8181	4	
: 4591	8182	4	
: 4592	8183	4	
: 4593	8184	3	
: 4594	8185	3	
: 4595	8186	2	

0000 00000 GET_EXPR:				WORD	Save nothing	
C6	AF	00	FB 00002	CALLS	#0, GET_TERM	8175
	29	50	E9 00006	BLBC	R0, 3\$	8179
		3C	DD 00009 1\$:	PUSHL	#6\$	8180
FF21	CF	01	FB 0000B	CALLS	#1, GET_KYW_TYPE_NM	
	1B	50	E9 00010	BLBC	R0, 2\$	
B5	AF	00	FB 00013	CALLS	#0, GET_TERM	8182
	18	50	E9 00017	BLBC	R0, 3\$	
50	54	04	A4 C1 0001A	ADDL3	4(LB), LB, R0	8183
		01	90 0001F	MOVB	#1, 20(R0)	
14	A0	04	A4 D6 00023	INCL	4(LB)	
		04	A4 D1 00026	CMPL	4(LB), #30	
	1E	DD	19 0002A	BLSS	1\$	
		04	11 0002C	BRB	3\$	
	50	01	D0 0002E 2\$:	MOVL	#1, R0	8185

50 04 00031 RET
04 00032 3\$ CLRL R0
04 00034 RET: 8186
:

: Routine Size: 53 bytes, Routine Base: SOR\$RO_CODE + 00CF

```
: 4596      8187  2
: 4597      8188  2
: 4598      8189  3
: 4599      8190  3
: 4600      8191  3
: 4601      8192  3
: 4602      8193  3
: 4603      8194  3
: 4604      8195  3
: 4605      8196  3
: 4606      8197  3
: 4607      8198  3
: 4608      8199  3
: 4609      8200  4
: 4610      8201  4
: 4611      8202  4
: 4612      8203  4
: 4613      8204  4
: 4614      8205  4
: 4615      8206  4
: 4616      8207  4
: 4617      8208  4
: 4618      8209  4
: 4619      8210  4
: 4620      8211  3
: 4621      8212  4
: 4622      8213  4
: 4623      8214  4
: 4624      8215  4
: 4625      8216  4
: 4626      8217  4
: 4627      8218  4
: 4628      8219  4
: 4629      8220  5
: 4630      8221  5
: 4631      8222  5
: 4632      8223  5
: 4633      8224  4
: 4634      8225  4
: 4635      8226  5
: 4636      8227  4
: 4637      8228  4
: 4638      8229  3
: 4639      8230  3
: 4640      8231  2

ROUTINE CEVAL(TLAB, FLAB): CA_LINKAGE_LB =
BEGIN
LOCAL
  SOP;          ! Saved operator
MACRO
  MO(X) = (IF X GTR 0 THEN X ELSE 0) %;
LB_DECL;
CA_AREA ( CA );
S2I = .S2I - 1;
SOP = .S2I[S2I];
IF .SOP EQL OP_AND OR .SOP EQL OP_OR
THEN
  BEGIN
    CEVAL(.TLAB, .FLAB);
    IF .SOP EQL OP_OR
      THEN CEVAL(.TLAB, .S1I)
      ELSE CEVAL(.S1I, .FLAB);
  END
%IF %DECLARED(OP_NOT) %THEN
ELSE IF .SOP EQL OP_NEG
THEN
  THEN CEVAL(.FLAB, .TLAB)
%FI
ELSE
  BEGIN
    LOCAL
      TDT: REF TDT_TAB[];
      TDT = .CA[CA_TDT_ADR];
      S1I = .S1I - 1;
      IF .TLAB EQL .S1I+1
      THEN
        BEGIN
          TDT[.S1I, TDT_CMP] = NOT .TDT[.S1I, TDT_CMP];
          TDT[.S1I, TDT_GOTO] = MO(.FLAB-.S1I);
        END
      ELSE IF .FLAB EQL .S1I+1
      THEN
        TDT[.S1I, TDT_GOTO] = MO(.TLAB-.S1I)
      ELSE
        RETURN FAIL;
  END;
RETURN SUCCESS;
END;
```


50		04	A4	D7	00002	CEVAL:	.WORD	Save R2,R3,R5	8188
	54	04	A4	C1	00005		DECL	4(LB)	8196
	53	14	A0	9A	0000A		ADDL3	4(LB), LB, R0	8197
	55	08	AC	D0	0000E		MOVZBL	20(R0), SOP	
	52	04	AC	D0	00012		MOVL	FLAB, R5	8201
			53	D5	00016		MOVL	TLAB, R2	
			05	13	00018		TSTL	SOP	8198
	01		53	D1	0001A		BEQL	1\$	
			21	12	0001D		CMPL	SOP, #1	
			24	BB	0001F	1\$:	BNEQ	4\$	
DB	AF		02	FB	00021		PUSHR	#*M<R2,R5>	8201
	01		53	D1	00025		CALLS	#2, CEVAL	
			09	12	00028		CMPL	SOP, #1	8202
	7E	0100	CB	9A	0002A		BNEQ	2\$	
			52	DD	0002F		MOVZBL	256(CA), -(SP)	8203
			07	11	00031		PUSHL	R2	
			55	DD	00033	2\$:	BRB	3\$	
	7E	0100	CB	9A	00035		PUSHL	R5	8204
C2	AF		02	FB	0003A	3\$:	MOVZBL	256(CA), -(SP)	
			4A	11	0003E		CALLS	#2, CEVAL	
	53	0114	CB	D0	00040	4\$:	BRB	10\$	8198
		0100	CB	97	00045		MOVL	276(CA), TDT	8215
	50	0100	CB	9A	00049		DECB	256(CA)	8217
	51	01	A0	9E	0004E		MOVZBL	256(CA), R0	8218
	51		52	D1	00052		MOVAB	1(R0), R1	
			1A	12	00055		CMPL	R2, R1	
	51	6340	DE	00057			BNEQ	7\$	
	61		0E	8C	0005B		MOVAL	(TDT)[R0], R1	8221
	50		55	D1	0005E		XORB2	#14, (R1)	
			06	15	00061		CMPL	R5, R0	8222
50	55		50	C3	00063		BLEQ	5\$	
			02	11	00067		SUBL3	R0, R5, R0	
			50	D4	00069	5\$:	BRB	6\$	
	03	A1	50	90	0006B	6\$:	CLRL	R0	
			19	11	0006F		MOVB	R0, 3(R1)	
	51		55	D1	00071	7\$:	BRB	10\$	8218
			18	12	00074		CMPL	R5, R1	8224
	50		52	D1	00076		BNEQ	11\$	
			06	15	00079		CMPL	R2, R0	8226
51	52		50	C3	0007B		BLEQ	8\$	
			02	11	0007F		SUBL3	R0, R2, R1	
			51	D4	00081	8\$:	BRB	9\$	
		03	A340	DF	00083	9\$:	CLRL	R1	
	9E		51	90	00087		PUSHAL	3(TDT)[R0]	
	50		01	D0	0008A	10\$:	MOVB	R1, @ (SP)+	
				04	0008D		MOVL	#1, R0	8230
			50	D4	0008E	11\$:	RET		
			04	00090			CLRL	R0	8231
							RET		

; Routine Size: 145 bytes, Routine Base: SOR\$RO_CODE + 0104

: 4641	8232	2	ROUTINE GET_SIMPLE: CA_LINKAGE_LB = BEGIN LOCAL
: 4642	8233	2	
: 4643	8234	3	
: 4644	8235	3	


```
: 4645      8236      TST: REF VECTOR[3],
: 4646      8237      TDT: REF TDT_TAB[1],
: 4647      8238      INDEX1,
: 4648      8239      INDEX2,
: 4649      8240      SUB_KYW_TYPE,
: 4650      8241      CON_SYM: REF CON_SYM_TAB[],
: 4651      8242      CON_CNT: REF VECTOR[1],
: 4652      8243      FLD_SYM: REF SYM_TAB[],
: 4653      8244      STRING: VECTOR[3];
: 4654      8245      LB_DECL;
: 4655      8246      CA_AREA_( CA );
: 4656      8247
: 4657      8248      TST      = .LB[K_TST];
: 4658      8249      CON_SYM = .LB[K_CON_SYM];
: 4659      8250      CON_CNT = .LB[K_CON_CNT];
: 4660      8251      FLD_SYM = .LB[K_FLD_SYM];
: 4661      8252
: 4662      8253      TDT = ALLOC (TDT_UNIT);
: 4663      8254      CA[CA_TDT_SIZ] = .CA[CA_TDT_SIZ] + 1;
: 4664      8255      IF .CA[CA_TDT_SIZ] GEQ TDT_MAX THEN RETURN FAIL;
: 4665      8256
: 4666      8257      ! Get field name
: 4667      8258      IF SKIP_IGNORED( TST[0] ) EQL FAIL THEN RETURN FAIL;
: 4668      8259      IF GET_STRING( TST[0], STRING[0] ) EQL FAIL THEN RETURN FAIL;
: 4669      8260
: 4670      8261      ! Search the field symbol table for a match
: 4671      8262      IF SEARCH_TABLE( STRING[0], INDEX1,
: 4672      8263          FLD_SYM[0, BASE_], .CA[CA_FDT_SIZ] ) EQL FAIL
: 4673      8264      THEN
: 4674      8265          RETURN FAIL;
: 4675      8266          ! Error if not found
: 4676      8267
: 4677      8268      ! Set field one index
: 4678      8269      TDT[0, TDT_FLD_ONE] = .FLD_SYM[.INDEX1, SYM_INDEX];
: 4679      8270
: 4680      8271      ! Clear the comparison flags, and the TRUE flag
: 4681      8272      TDT[0, TDT_EQL] = FALSE;
: 4682      8273      TDT[0, TDT_LSS] = FALSE;
: 4683      8274      TDT[0, TDT_GTR] = FALSE;
: 4684      8275      TDT[0, TDT_TRUE] = FALSE;
: 4685      8276
: 4686      8277      ! Get and parse the operator keyword
: 4687      8278      IF SKIP_IGNORED( TST[0] ) EQL FAIL THEN RETURN FAIL;
: 4688      8279      SUB_KYW_TYPE = GET KYW TYPE( TST[0] );
: 4689      8280      SELECT .SUB_KYW_TYPE OF
: 4690      8281          SET
: 4691      8282          [KW_EQ, KW_LE, KW_GE]:      TDT[0, TDT_EQL] = TRUE;
: 4692      8283          [KW_LT, KW_LE, KW_NE]:      TDT[0, TDT_LSS] = TRUE;
: 4693      8284          [KW_GT, KW_GE, KW_NE]:      TDT[0, TDT_GTR] = TRUE;
: 4694      8285          [OTHERWISE]:      RETURN FAIL;
: 4695      8286      TES;
: 4696      8287
: 4697      8288      ! Get the constant or second field
: 4698      8289      IF SKIP_IGNORED( TST[0] ) EQL FAIL THEN RETURN FAIL;
: 4699      8290      IF GET_STRING( TST[0], STRING[0] ) EQL FAIL THEN RETURN FAIL;
: 4700      8291
: 4701      8292      ! Search the field symbol table for a match
```


4702 8293 3
4703 8294 3
4704 8295 3
4705 8296 4
4706 8297 4
4707 8298 4
4708 8299 4
4709 8300 4
4710 8301 4
4711 8302 4
4712 8303 4
4713 8304 4
4714 8305 3
4715 8306 4
4716 8307 4
4717 8308 4
4718 8309 4
4719 8310 4
4720 8311 4
4721 8312 4
4722 8313 4
4723 8314 4
4724 8315 4
4725 8316 4
4726 8317 4
4727 8318 4
4728 8319 4
4729 8320 4
4730 8321 4
4731 8322 4
4732 8323 4
4733 8324 4
4734 8325 4
4735 8326 4
4736 8327 4
4737 8328 4
4738 8329 4
4739 8330 3
4740 8331 3
4741 8332 2

```
IF SEARCH_TABLE( STRING[0], INDEX2,  
    FLD_SYM[0,BASE_], .CALCA_FDT_SIZE ) EQL SUCCESS  
THEN  
    BEGIN  
        We have a field (not a constant)  
        Indicate it is not a constant, and store the field two index  
        TDT[0, TDT_CONSTANT] = FALSE;  
        TDT[0, TDT_FLD_TWO] = .FLD_SYM[INDEX2, SYM_INDEX];  
    END  
ELSE  
    BEGIN  
        We have a constant. Make an entry in the constant symbol table  
        IF .CON_CNT[0] GEQ MAX_CONST THEN RETURN FAIL;      ! too many?  
        Put index to constant table in TDT, and mark as a constant  
        TDT[0, TDT_FLD_TWO] = .CON_CNT[0];  
        TDT[0, TDT_CONSTANT] = TRUE;  
        Initialize the constant table entry  
        Store the length and address of the constant  
        Store the field one index. This indicates a related field,  
        and lets us know how to convert this thing later.  
        Zero the length field.  
        CON_SYM[CON_CNT[0], CON_NAM_LEN] = .STRING_LEN;  
        CON_SYM[CON_CNT[0], CON_NAM_ADR] = .STRING_PTR;  
        CON_SYM[CON_CNT[0], CON_INDEX] = .FLD_SYM[INDEX1, SYM_INDEX];  
        CON_SYM[CON_CNT[0], CON_LENGTH] = 0;  
        CON_CNT[0] = .CON_CNT[0] + 1;  
    END;  
RETURN SUCCESS;  
END;
```

07EC 00000 GET_SIMPLE:						
5A	00000000G	00	9E 00002	.WORD	Save R2,R3,R5,R6,R7,R8,R9,R10	8233
59	00000000G	00	9E 00009	MOVAB	GSSGET STRING, R10	
5E		14	C2 00010	MOVAB	SI\$SKIP_IGNORED, R9	
56		64	D0 00013	SUBL2	#20, SP	8248
58	08	A4	D0 00016	MOVL	(LB), TST	8249
57	0C	A4	D0 0001A	MOVL	8(LB), CON_SYM	8250
53	10	A4	D0 0001E	MOVL	12(LB), CON_CNT	8251
		04	DD 00022	MOVL	16(LB), FLD_SYM	8253
00000000G	00	01	FB 00024	PUSHL	#4	
	52	50	D0 0002B	CALLS	#1, SASSPC_ALLOC	
				MOVL	R0, TDT	

FF	8F	0100	CB	96	0002E	INCB	256(CA)	8254
		0100	CB	91	00032	CMPB	256(CA), #255	8255
			03	1F	00038	BLSSU	2\$	
			00F6	31	0003A	BRW	10\$	
			56	DD	0003D	PUSHL	TST	8258
	69		01	FB	0003F	CALLS	#1, SI\$SKIP_IGNORED	
			50	D5	00042	TSTL	R0	
			F4	13	00044	BEQL	1\$	
		08	AE	9F	00046	PUSHAB	STRING	8259
			56	DD	00049	PUSHL	TST	
	6A		02	FB	0004B	CALLS	#2, GS\$GET_STRING	
			50	D5	0004E	TSTL	R0	
			E8	13	00050	BEQL	1\$	
	7E	00FF	CB	9A	00052	MOVZBL	255(CA), -(SP)	8263
			53	DD	00057	PUSHL	FLD_SYM	
		08	AE	9F	00059	PUSHAB	INDEX1	8262
		14	AE	9F	0005C	PUSHAB	STRING	
00000000G	00		04	FB	0005F	CALLS	#4, ST\$SEARCH_TABLE	8263
			50	D5	00066	TSTL	R0	
			D0	13	00068	BEQL	1\$	
55			06	C5	0006A	MULL3	#6, INDEX1, R5	8268
	01	6E	05	A543	90	MOV	5(R5)[FLD_SYM], 1(TDT)	
		A2		0F	8A	BICB2	#15, (TDT)	8274
		62		56	DD	PUSHL	TST	8277
				01	FB	CALLS	#1, SI\$SKIP_IGNORED	
		69		50	D5	TSTL	R0	
				BA	13	BEQL	1\$	
				56	DD	PUSHL	TST	8278
00000000G	00		01	FB	00082	CALLS	#1, GKT\$GET_KYW_TYPE	
	51		01	D0	00089	MOVL	#1, R1	8279
	35		50	D1	0008C	CMPL	SUB_KYW_TYPE, #53	8281
			0A	13	0008F	BEQL	3\$	
	37		50	D1	00091	CMPL	SUB_KYW_TYPE, #55	
			05	13	00094	BEQL	3\$	
	39		50	D1	00096	CMPL	SUB_KYW_TYPE, #57	
			05	12	00099	BNEQ	4\$	
			51	D4	0009B	CLRL	R1	
	62		02	88	0009D	BISB2	#2, (TDT)	
	36		50	D1	000A0	CMPL	SUB_KYW_TYPE, #54	8282
			0A	19	000A3	BLSS	5\$	
	38		50	D1	000A5	CMPL	SUB_KYW_TYPE, #56	
			05	14	000A8	BGTR	5\$	
			51	D4	000AA	CLRL	R1	
	62		04	88	000AC	BISB2	#4, (TDT)	
	36		50	D1	000AF	CMPL	SUB_KYW_TYPE, #54	8283
			0A	13	000B2	BEQL	6\$	
	39		50	D1	000B4	CMPL	SUB_KYW_TYPE, #57	
			0A	19	000B7	BLSS	7\$	
	3A		50	D1	000B9	CMPL	SUB_KYW_TYPE, #58	
			05	14	000BC	BGTR	7\$	
			51	D4	000BE	CLRL	R1	
	62		08	88	000C0	BISB2	#8, (TDT)	
	6D		51	E8	000C3	BLBS	R1, 10\$	8284
			56	DD	000C6	PUSHL	TST	8288
			01	FB	000C8	CALLS	#1, SI\$SKIP_IGNORED	
	69		50	D5	000CB	TSTL	R0	
			64	13	000CD	BEQL	10\$	

		08	AE	9F	000CF	PUSHAB	STRING	8289	
			56	DD	000D2	PUSHL	TST		
	6A		02	FB	000D4	CALLS	#2, GS\$GET_STRING		
			50	D5	000D7	TSTL	R0		
			58	13	000D9	BEQL	10\$		
	7E	00FF	CB	9A	000DB	MOVZBL	255(CA), -(SP)	8294	
			53	DD	000E0	PUSHL	FLD_SYM		
		0C	AE	9F	000E2	PUSHAB	INDEX2	8293	
		14	AE	9F	000E5	PUSHAB	STRING		
00000000G	00		04	FB	000E8	CALLS	#4, ST\$SEARCH_TABLE	8294	
	01		50	D1	000EF	CMPL	R0, #1		
			10	12	000F2	BNEQ	8\$		
50	62		10	8A	000F4	BICB2	#16, (TDT)	8302	
	04	AE	06	C5	000F7	MULL3	#6, INDEX2, R0	8303	
	02	A2	05	A043	90	000FC	MOVB	5(R0)[FLD_SYM], 2(TDT)	
			28	11	00102	BRB	9\$	8293	
	50		67	D0	00104	MOVL	(CON_CNT), R0	8310	
000000FF	8F		50	D1	00107	CMPL	R0, #255		
			23	18	0010E	BGEQ	10\$		
	02	A2	50	90	00110	MOVB	R0, 2(TDT)	8314	
	62		10	88	00114	BISB2	#16, (TDT)	8315	
	50		6840	7E	00117	MOVAQ	(CON_SYM)[R0], R0	8324	
	04	A0	08	AE	90	0011B	MOVB	STRING, 4(R0)	
	60		0C	AE	D0	00120	MOVL	STRING+4, (R0)	8325
	05	A0	05	A543	90	00124	MOVB	5(R5)[FLD_SYM], 5(R0)	8326
			06	A0	94	0012A	CLRB	6(R0)	8327
			67	D6	0012D	INCL	(CON_CNT)	8328	
	50		01	D0	0012F	MOVL	#1, R0	8331	
				04	00132	RET			
			50	D4	00133	CLRL	R0	8332	
				04	00135	RET			

; Routine Size: 310 bytes, Routine Base: SOR\$RO_CODE + 0195

4742	8333	2	
4743	8334	2	LOCAL
4744	8335	2	S1S, ! Saved value of S1I
4745	8336	2	LBLOCK: VECTOR[K_S2+S2_MAX+1];
4746	8337	2	
4747	8338	2	GLOBAL REGISTER LB = LB_REG: REF VECTOR;
4748	8339	2	
4749	8340	2	CA_AREA_(CA);
4750	8341	2	
4751	8342	2	output_('parse_test');
4752	8343	2	
4753	8344	2	LB = LBLOCK[0];
4754	8345	2	LB[K_S2I] = 0;
4755	8346	2	LB[K_TST] = TST[0];
4756	8347	2	LB[K_CON_SYM] = CON_SYM[0, BASE_];
4757	8348	2	LB[K_CON_CNT] = CON_CNT[0];
4758	8349	2	LB[K_FLD_SYM] = FLD_SYM[0, BASE_];
4759	8350	2	
4760	8351	2	IF NOT GET_EXPR() THEN RETURN FAIL;
4761	8352	2	IF SKIP_IGNORED(TST[0]) NEQ FAIL THEN RETURN FAIL;
4762	8353	2	
4763	8354	2	S1S = .S1I;

```
: 4764      8355 2      IF NOT CEVAL(.S11, 0) THEN RETURN FAIL;
: 4765      8356 2      S11 = .S1S;
: 4766      8357 2
: 4767      8358 2      ! Create one more entry that has the TRUE bit set
: 4768      8359 2      !
: 4769      8360 2      BEGIN
: 4770      8361 2      LOCAL
: 4771      8362 2          TDT: REF TDT_TAB[1];
: 4772      8363 2          TDT = ALLOC (TDT_UNIT);
: 4773      8364 2          CALCA TDT_SIZE = CALCA TDT_SIZE + 1;
: 4774      8365 2          IF CALCA TDT_SIZE GEQ TDT_MAX THEN RETURN FAIL;
: 4775      8366 2          TDT[0, TDT_TRUE] = TRUE;
: 4776      8367 2          END;
: 4777      8368 2
: 4778      8369 2      RETURN SUCCESS
: 4779      8370 1      END;
```

			0014 00000	.ENTRY	PT\$PARSE_TEST, Save R2,R4		8031
	5E	FF70	CE 9E 00002	MOVAB	-144(SP), SP		
	54		6E 9E 00007	MOVAB	LBLOCK, LB		8344
		04	A4 D4 0000A	CLRL	4(LB)		8345
	64	04	AC D0 0000D	MOVL	TST, (LB)		8346
08	A4	08	AC 7D 00011	MOVQ	CON_SYM, 8(LB)		8347
10	A4	10	AC D0 00016	MOVL	FLD_SYM, 16(LB)		8349
FDE4	CF		00 FB 0001B	CALLS	#0, GET_EXPR		8351
	43		50 E9 00020	BLBC	R0, 1\$		
		04	AC DD 00023	PUSHL	TST		8352
00000000G	00		01 FB 00026	CALLS	#1, SI\$SKIP_IGNORED		
			50 D5 0002D	TSTL	R0		
			35 12 0002F	BNEQ	1\$		
	52	0100	CB 9A 00031	MOVZBL	256(CA), S1S		8354
			7E D4 00036	CLRL	-(SP)		8355
	7E	0100	CB 9A 00038	MOVZBL	256(CA), -(SP)		
FDF7	CF		02 FB 0003D	CALLS	#2, CEVAL		
	21		50 E9 00042	BLBC	R0, 1\$		
0100	CB		52 90 00045	MOVB	S1S, 256(CA)		8356
			04 DD 0004A	PUSHL	#4		8363
00000000G	00		01 FB 0004C	CALLS	#1, SA\$SPC_ALLOC		
		0100	CB 96 00053	INCB	256(CA)		8364
FF	8F	0100	CB 91 00057	CMPB	256(CA), #255		8365
			07 1E 0005D	BGEQU	1\$		
	60		01 88 0005F	BISB2	#1, (TDT)		8366
	50		01 D0 00062	MOVL	#1, R0		8369
			04 00065	RET			
			50 D4 00066 1\$:	CLRL	R0		8370
			04 00068	RET			

; Routine Size: 105 bytes, Routine Base: SOR\$RO_CODE + 02CB

```
: 4780      8371 1
: 4781      8372 1
: 4782      8373 1 ! ELUDOM
```


PT\$PAR
T03-038

H 10
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 160
(34)

: 4783 8374 1 !
: 4784 8375 1 END
: 4785 8376 0 ELUDOM

PSECT SUMMARY

: Name Bytes Attributes
: SOR\$RO_CODE 820 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	103	25	34	00:00.1
-\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	24	20	12	00:00.1
-\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI
:)

: 4786 L 8377 0 MODULE P\$PAR (IDENT = 'T03-038' %REQUIRE('SRC\$:DEFS0')) =
: 4787 8383 1 BEGIN
: 4788 8384 1 REQUIRE 'SRC\$:DKS';
: 4789 8682 1 EXTERNAL ROUTINE
: 4790 8683 1 ERR_CNTL: CA_LINKAGE, ! Error control routine
: 4791 8684 1 COL\$MODIFY, ! Make a modification
: 4792 8685 1 SKIP_COMMA, ! Skip commas and ignored stuff
: 4793 8686 1 GET_CHAR_CLAUSE; ! Get character clauses of various kinds

```
: 4795      8687 1 GLOBAL ROUTINE PARSE IGN(  
: 4796      8688 1     IGN: REF VECTOR[3],  
: 4797      8689 1     CD:  REF VECTOR[2] ): CA_LINKAGE =  
: 4798      8690 1  
: 4799      8691 1 +-  
: 4800      8692 1 | Functional Description:  
: 4801      8693 1 |  
: 4802      8694 1 |     Parse an ignore clause  
: 4803      8695 1 |  
: 4804      8696 1 | Formal Parameters:  
: 4805      8697 1 |  
: 4806      8698 1 |     IGN      - Address of length/address/line of the ignore clause.  
: 4807      8699 1 |     CD       - Length/address of the collating description work area  
: 4808      8700 1 |  
: 4809      8701 1 | Implicit Inputs:  
: 4810      8702 1 |  
: 4811      8703 1 | Implicit Outputs:  
: 4812      8704 1 |  
: 4813      8705 1 | Routine Value:  
: 4814      8706 1 |  
: 4815      8707 1 |     SUCCESS  
: 4816      8708 1 |     FAIL  
: 4817      8709 1 |  
: 4818      8710 1 | Side Effects:  
: 4819      8711 1 |  
: 4820      8712 1 | --  
: 4821      8713 2 | BEGIN  
: 4822      8714 2 |  
: 4823      8715 2 | LOCAL  
: 4824      8716 2 |     CH4: VECTOR[4,BYTE], ! Characters returned from get_char_clause  
: 4825      8717 2 |     INDEX,  
: 4826      8718 2 |     CLAUSE_TYPE;  
: 4827      8719 2 | LOCAL  
: 4828      8720 2 |     KYW_LINE,  
: 4829      8721 2 |     CHAR1: VECTOR[4,BYTE],  
: 4830      8722 2 |     CHAR2: VECTOR[4,BYTE],  
: 4831      8723 2 |     S;  
: 4832      8724 2 |  
: 4833      8725 2 | CA_AREA_( CA );  
: 4834      8726 2 |  
: 4835      8727 2 | output_('parse_ign');  
: 4836      8728 2 |  
: 4837      8729 2 | KYW_LINE = 0;  
: 4838      8730 2 |  
: 4839      8731 2 | CHAR1[0] = 0;  
: 4840      8732 2 | CHAR1[1] = 0;  
: 4841      8733 2 | CHAR2[0] = 0;  
: 4842      8734 2 | CHAR2[1] = 0;  
: 4843      8735 2 |  
: 4844      8736 2 | ! While there are more ignore sub-clauses  
: 4845      8737 2 | WHILE TRUE DO  
: 4846      8738 2 |     BEGIN  
: 4847      8739 2 |  
: 4848      8740 2 |         ! Get character clause  
: 4849      8741 2 |         CLAUSE_TYPE = GET_CHAR_CLAUSE( IGN[0], CH4[0] );  
: 4850      8742 2 |  
: 4851      8743 3 |     SELECTONE .CLAUSE_TYPE OF
```


4852 8744 3
4853 8745 3
4854 8746 3
4855 8747 4
4856 8748 4
4857 8749 4
4858 8750 5
4859 8751 5
4860 8752 5
4861 8753 5
4862 8754 4
4863 8755 3
4864 8756 3
4865 8757 3
4866 8758 4
4867 8759 4
4868 8760 4
4869 8761 4
4870 8762 4
4871 8763 4
4872 8764 3
4873 8765 3
4874 8766 3
4875 8767 3
4876 8768 3
4877 8769 3
4878 8770 3
4879 8771 3
4880 8772 2
4881 8773 2
4882 8774 2
4883 8775 2
4884 8776 1

```
SET
[GC_SINGLE, GC_S_TO_S]:      ! Single or range
  BEGIN
  CHAR1[0] = 1;
  INCRU I FROM .CH4[0] TO .CH4[2] DO
    BEGIN
    CHAR1[2] = .I;
    S = COLL$MODIFY(CD[0], CHAR1, CHAR2, 0);
    IF ERROR_(.S) THEN (ERR_(.KYW_LINE, .S); RETURN FAIL);
    END;
  END;
[GC_DOUBLE]:
  BEGIN
  CHAR1[0] = 2;
  CHAR1[2] = .CH4[0];
  CHAR1[3] = .CH4[1];
  S = COLL$MODIFY(CD[0], CHAR1, CHAR2, 0);
  IF ERROR_(.S) THEN (ERR_(.KYW_LINE, .S); RETURN FAIL);
  END;
[OTHERWISE]:                  ! Error condition
  RETURN FAIL;
TES;
IF SKIP_COMMA( IGN[0] ) EQL FAIL THEN EXITLOOP;
END;
RETURN SUCCESS
END;
```

				.TITLE	PISPAR	
				.IDENT	\T03-038\	
				.EXTRN	EC\$ERR CNTL, COLL\$MODIFY	
				.EXTRN	SC\$SKIP_COMMA, GCC\$GET_CHAR_CLAUSE	
				.PSECT	SOR\$RO_CODE, NOWRT, SHR, PIC, 2	
				.ENTRY	PISPARSE IGN, Save R2, R3, R4, R5, R6, R7	8687
57	00000000G	00	9E	MOVAB	COLL\$MODIFY, R7	
5E		0C	C2	SUBL2	#12, SP	
		55	D4	CLRL	KYW_LINE	8729
	08	AE	B4	CLRW	CHAR1	8731
	04	AE	B4	CLRW	CHAR2	8733
		5E	DD	PUSHL	SP	8741
		04	AC	PUSHL	IGN	
00000000G	00	02	FB	CALLS	#2, GCC\$GET_CHAR_CLAUSE	
	56	50	D0	MOVL	R0, CLAUSE_TYPE	
	01	56	D1	CMPL	CLAUSE_TYPE, #1	8746
		05	13	BEQL	2\$	
	03	56	D1	CMPL	CLAUSE_TYPE, #3	
		2E	12	BNEQ	5\$	

08	AE	02	01	90	0002D	2\$:	MOVB	#1, CHAR1	:	8748
	53		AE	9A	00031		MOVZBL	CH4+2, R3	:	8749
	52		6E	9A	00035		MOVZBL	CH4, 1	:	8752
			1A	11	00038		BRB	4\$	:	
0A	AE		52	90	0003A	3\$:	MOVB	I, CHAR1+2	:	8751
			7E	D4	0003E		CLRL	-(SP)	:	8752
		08	AE	9F	00040		PUSHAB	CHAR2	:	
		10	AE	9F	00043		PUSHAB	CHAR1	:	
		08	AC	DD	00046		PUSHL	CD	:	
	67		04	FB	00049		CALLS	#4, COLL\$MODIFY	:	
	54		50	D0	0004C		MOVL	R0, S	:	
	2A		54	E9	0004F		BLBC	S, 6\$	:	8753
			52	D6	00052		INCL	I	:	8749
	53		52	D1	00054	4\$:	CMPL	I, R3	:	
			E1	1B	00057		BLEQU	3\$	:	
			2E	11	00059		BRB	7\$	:	8743
	02		56	D1	0005B	5\$:	CMPL	CLAU\$E_TYPE, #2	:	8757
			3E	12	0005E		BNEQ	9\$	:	
08	AE		02	90	00060		MOVB	#2, CHAR1	:	8759
0A	AE		6E	B0	00064		MOVW	CH4, CHAR1+2	:	8760
			7E	D4	00068		CLRL	-(SP)	:	8762
		08	AE	9F	0006A		PUSHAB	CHAR2	:	
		10	AE	9F	0006D		PUSHAB	CHAR1	:	
		08	AC	DD	00070		PUSHL	CD	:	
	67		04	FB	00073		CALLS	#4, COLL\$MODIFY	:	
	54		50	D0	00076		MOVL	R0, S	:	
	0D		54	E8	00079		BLBS	S, 7\$	:	8763
			54	DD	0007C	6\$:	PUSHL	S	:	
			55	DD	0007E		PUSHL	KYW_LINE	:	
00000000G	00		02	FB	00080		CALLS	#2, EC\$ERR_CNTL	:	
			15	11	00087		BRB	9\$	:	
		04	AC	DD	00089	7\$:	PUSHL	IGN	:	8771
00000000G	00		01	FB	0008C		CALLS	#1, SC\$SKIP_COMMA	:	
			50	D5	00093		TSTL	R0	:	
			03	13	00095		BEQL	8\$	:	
			FF7A	31	00097		BRW	1\$	:	
	50		01	D0	0009A	8\$:	MOVL	#1, R0	:	8774
				04	0009D		RET		:	
			50	D4	0009E	9\$:	CLRL	R0	:	8776
				04	000A0		RET		:	

; Routine Size: 161 bytes, Routine Base: SOR\$RO_CODE + 0000

: 4885	8777	1	
: 4886	8778	1	
: 4887	8779	1	! ELUDOM
: 4888	8780	1	!
: 4889	8781	1	END
: 4890	8782	0	ELUDOM

PSECT SUMMARY


```

:      Name                      Bytes          Attributes
:  SOR$RO_CODE                  161  NOVEC,NOWRT, RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)

```

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	102	24	34	00:00.1
_\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	6	5	12	00:00.1
_\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI)

```

: 4891      L 8783 0 MODULE PM$PAR  (IDENT = 'T03-038' %REQUIRE('SRC$:DEFS0') ) =
: 4892      8789 1 BEGIN
: 4893      8790 1 REQUIRE 'SRC$:DKS';
: 4894      9088 1 EXTERNAL ROUTINE
: 4895      9089 1      ERR_CNTL:      CA_LINKAGE,      ! Error control routine
: 4896      9090 1      COL$MODIFY,      ! Make a modification
: 4897      9091 1      SKIP_COMMA,      ! Skip commas and ignored stuff
: 4898      9092 1      GET_CHAR_CLAUSE;      ! Get character clauses of various kinds

```

```

: 4900      9093 1 GLOBAL ROUTINE PARSE_MOD(
: 4901      9094 1     MODI: REF VECTOR[3],
: 4902      9095 1     CD: REF VECTOR[2]): CA_LINKAGE =
: 4903      9096 1
: 4904      9097 1 ++
: 4905      9098 1   Functional Description:
: 4906      9099 1     Parse a modification clause
: 4907      9100 1
: 4908      9101 1   Formal Parameters:
: 4909      9102 1
: 4910      9103 1     MODI - Address of length/address/line of the modification clause.
: 4911      9104 1     CD  - Length/address of the collating description work area
: 4912      9105 1
: 4913      9106 1   Implicit Inputs:
: 4914      9107 1
: 4915      9108 1   Implicit Outputs:
: 4916      9109 1
: 4917      9110 1   Routine Value:
: 4918      9111 1
: 4919      9112 1     SUCCESS
: 4920      9113 1     FAIL
: 4921      9114 1
: 4922      9115 1   Side Effects:
: 4923      9116 1
: 4924      9117 1 --
: 4925      9118 1
: 4926      9119 2   BEGIN
: 4927      9120 2
: 4928      9121 2   LOCAL
: 4929      9122 2     CH4: VECTOR[4,BYTE], ! Characters returned from get_char_clause
: 4930      9123 2     CLAUSE_TYPE;
: 4931      9124 2   LOCAL
: 4932      9125 2     KYW_LINE,
: 4933      9126 2     CHAR1: VECTOR[4,BYTE],
: 4934      9127 2     CHAR2: VECTOR[4,BYTE],
: 4935      9128 2     S;
: 4936      9129 2
: 4937      9130 2   CA_AREA_( CA );
: 4938      9131 2
: 4939      9132 2   output_('parse_mod');
: 4940      9133 2
: 4941      9134 2   KYW_LINE = 0;
: 4942      9135 2
: 4943      9136 2   CHAR1[1] = 0;
: 4944      9137 2   CHAR2[1] = 0;
: 4945      9138 2
: 4946      9139 2   ! While there are more modification sub-clauses
: 4947      9140 2   WHILE TRUE DO
: 4948      9141 2     BEGIN
: 4949      9142 2
: 4950      9143 2       ! Find type of modification
: 4951      9144 2       CLAUSE_TYPE = GET_CHAR_CLAUSE( MODI[0], CH4[0] );
: 4952      9145 2
: 4953      9146 2       SELECTONE .CLAUSE_TYPE OF
: 4954      9147 2         SET
: 4955      9148 2         [FAIL, GC_SINGLE, GC_DOUBLE, GC_S_TO_S]: RETURN FAIL;
: 4956      9149 2         [OTHERWISE]: 0;
```


4957 9150
4958 9151
4959 9152
4960 9153
4961 9154
4962 9155
4963 9156
4964 9157
4965 9158
4966 9159
4967 9160
4968 9161
4969 9162
4970 9163
4971 9164
4972 9165
4973 9166

TES;

CHAR1[0] = GC L1 (.CLAUSE_TYPE);
CHAR1[2] = .CH4[0];
CHAR1[3] = .CH4[1];
CHAR2[0] = GC L2 (.CLAUSE_TYPE);
CHAR2[2] = .CH4[2];
CHAR2[3] = .CH4[3];

S = COLL\$MODIFY(CD[0], CHAR1, CHAR2, GC OP (.CLAUSE_TYPE));
IF_ERROR(.S) THEN (ERR(.KYW_LINE, .S);-RETURN FAIL);

IF SKIP_COMMA(MODI[0]) EQL FAIL THEN EXITLOOP;
END;

RETURN SUCCESS;
END;

5E 001C 00000
0C C2 00002
54 D4 00005
09 AE 94 00007
05 AE 94 0000A
5E DD 0000D 1\$:
04 AC DD 0000F
00000000G 00
52 02 FB 00012
05 50 D0 00019
03 05 19 0001C
52 D1 0001E
55 15 00021
50 52 04 EF 00023 2\$:
08 AE 50 90 00028
0A AE 6E B0 0002C
50 00 EF 00030
04 AE 50 90 00035
06 AE 02 AE B0 00039
02 02 EE 0003E
08 AE 9F 00043
10 AE 9F 00046
08 AC DD 00049
04 FB 0004C
00000000G 00
53 50 D0 00053
0D 53 E8 00056
53 DD 00059
54 DD 0005B
00000000G 00
02 FB 0005D
12 11 00064
04 AC DD 00066 3\$:

.TITLE PM\$PAR
.IDENT \T03-038\
.EXTRN EC\$ERR_CNTL, COLL\$MODIFY
.EXTRN SC\$SKIP_COMMA, GCC\$GET_CHAR_CLAUSE
.PSECT SOR\$RO_CODE,NOWRT, SHR, PIC,2

.ENTRY PM\$PARSE_MOD, Save R2,R3,R4 : 9093
SUBL2 #12, SP
CLRL KYW_LINE : 9134
CLRB CHAR1+1 : 9136
CLRB CHAR2+1 : 9137
PUSHL SP : 9144
PUSHL MODI
CALLS #2, GCC\$GET_CHAR_CLAUSE
MOVL R0, CLAUSE_TYPE
BLSS 2\$: 9148
CMLP CLAUSE_TYPE, #3
BLEQ 4\$
EXTZV #4, #2, CLAUSE_TYPE, R0 : 9152
MOVB R0, CHAR1
MOVW CH4, CHAR1+2 : 9153
EXTZV #0, #2, CLAUSE_TYPE, R0 : 9155
MOVB R0, CHAR2
MOVW CH4+2, CHAR2+2 : 9156
EXTV #2, #2, CLAUSE_TYPE, -(SP) : 9159
PUSHAB CHAR2
PUSHAB CHAR1
PUSHL CD
CALLS #4, COLL\$MODIFY
MOVL R0, S
BLBS S, 3\$: 9160
PUSHL S
PUSHL KYW_LINE
CALLS #2, EC\$ERR_CNTL
BRB 4\$
PUSHL MODI : 9162

00000000G 00

01 FB 00069
50 D5 00070
99 12 00072
01 D0 00074
04 00077
50 D4 00078 4\$:
04 0007ACALLS #1, SC\$SKIP_COMMA
TSTL R0
BNEQ 1\$
MOVL #1, R0
RET
CLRL R0
RET: 9165
: 9166
:

: Routine Size: 123 bytes, Routine Base: SOR\$RO_CODE + 0000

: 4974 9167 1
: 4975 9168 1
: 4976 9169 1 ! ELUDOM
: 4977 9170 1 !
: 4978 9171 1 END
: 4979 9172 0 ELUDOM

PSECT SUMMARY

: Name Bytes Attributes
: SOR\$RO_CODE 123 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

: File Total Symbols Loaded Percent Pages Mapped Processing Time
: _\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1 409 102 24 34 00:00.1
: _\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1 120 6 5 12 00:00.1
: _\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1 1 1 100 9 00:00.1

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI
:): 4980 L 9173 0 MODULE CC\$CON (IDENT = 'T03-038' %REQUIRE('SRC\$:DEFS0')) =
: 4981 9179 1 BEGIN
: 4982 9180 1 REQUIRE 'SRC\$:DKS';
: 4983 9478 1 EXTERNAL ROUTINE

CC\$CON
T03-038

: 4984
: 4985

9479 1
9480 1

ERR_CNTL:
SPC_ALLOC:

CA_LINKAGE,
CA_LINKAGE;

C 11
16-Sep-1984 00:53:54
14-Sep-1984 13:10:54

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SRTSPCBLI.B32;1

Page 168
(36)

! Error control routine
! Allocate storage from the work area

```
4987 9481 1 GLOBAL ROUTINE CONV CONSTANTS(
4988 9482 1     FDT      : REF FDT_TAB[],
4989 9483 1     CFT      : REF CFT_TAB[],
4990 9484 1     CON_SYM  : REF CON_SYM_TAB[],
4991 9485 1     CON_CNT
4992 9486 1 ): CA_LINKAGE =
4993 9487 1
4994 9488 1 ++
4995 9489 1 Functional Description:
4996 9490 1     Convert the constant symbol table into the constant field table
4997 9491 1
4998 9492 1 Formal Parameters:
4999 9493 1
5000 9494 1     FDT      - Address of the field definition table
5001 9495 1     CFT      - Address of constant field table
5002 9496 1     CON_SYM  - Address of constant symbol table
5003 9497 1     CON_CNT  - Number of entries in constant symbol table
5004 9498 1
5005 9499 1 Implicit Inputs:
5006 9500 1
5007 9501 1 Implicit Outputs:
5008 9502 1
5009 9503 1 Routine Value:
5010 9504 1
5011 9505 1     SUCCESS
5012 9506 1     FAIL
5013 9507 1
5014 9508 1 Side Effects:
5015 9509 1
5016 9510 1 Note:
5017 9511 1     Mandatory length as used below is the minimum length required
5018 9512 1     by a conditional key specification.  If the CON_LENGTH field is
5019 9513 1     zero then the constant was not defined by a conditional key
5020 9514 1     specification, but rather by a condition specification.  In that
5021 9515 1     case, the length of the constant field should be determined by
5022 9516 1     the length of the field used in the condition specification.
5023 9517 1
5024 9518 1 --
5025 9519 2 BEGIN
5026 9520 2
5027 9521 2 LOCAL
5028 9522 2     OUT_PTR:      REF VECTOR[BYTE],      ! Output constant area
5029 9523 2     IN_CON_PTR,   ! Input constant pointer
5030 9524 2     IN_CON_LEN,   ! Input constant length
5031 9525 2     OUT_CON_IDX,  ! Total output constant length
5032 9526 2     PRE_LENGTH;   ! Presumed length
5033 9527 2
5034 9528 2 ! Get the impure data area
5035 9529 2 CA_AREA_( CA );
5036 9530 2
5037 9531 2 output_('conv_constants');
5038 9532 2
5039 9533 2 ! Set the address of constant storage area behind the CFT
5040 9534 2 ! Set the address of constant area in common area
5041 9535 2
5042 9536 2 CA[CA_CONST_AREA] = ALLOC_(0);
5043 9537 2
```



```
5044 9538 2
5045 9539 2
5046 9540 2
5047 9541 2
5048 9542 2
5049 9543 2
5050 9544 2
5051 9545 2
5052 9546 2
5053 9547 2
5054 9548 2
5055 9549 2
5056 9550 2
5057 9551 2
5058 9552 2
5059 9553 2
5060 9554 2
5061 9555 2
5062 9556 2
5063 9557 2
5064 9558 2
5065 9559 4
5066 9560 4
5067 L 9561 4
5068 9562 4
5069 9563 4
5070 9564 4
5071 9565 4
5072 9566 4
5073 9567 4
5074 9568 4
5075 9569 4
5076 9570 4
5077 9571 4
5078 9572 4
5079 9573 4
5080 9574 4
5081 9575 4
5082 9576 4
5083 9577 4
5084 9578 4
5085 9579 4
5086 9580 4
5087 9581 4
5088 9582 4
5089 9583 4
5090 9584 4
5091 9585 4
5092 9586 4
5093 9587 4
5094 9588 4
5095 9589 4
5096 9590 4
5097 9591 4
5098 9592 4
5099 9593 4
5100 9594 3
```

```
! Set up the constant table.
! Run through the constant symbol table and convert
INCR IN FROM 0 TO .CON_CNT-1 DO
  BEGIN
  BIND
    WF_NAMES = CA[CA_WF_NAMES]: VECTOR[,BYTE];
  MACRO
    ITS_A_WORK_FILE = (.IN - .WF_NAMES[1] LSSU .WF_NAMES[0]) %;

    ! Get the start address and length of the constant
    ! Get the presumed length from the constant symbol table

    IN_CON_PTR = .CON_SYM[.IN, CON_NAM_ADR];
    IN_CON_LEN = .CON_SYM[.IN, CON_NAM_LEN];
    PRE_LENGTH = .CON_SYM[.IN, CON_LENGTH];

    ! If there, get length (in bytes) out of the FDT table
    IF .CON_SYM[.IN, CON_INDEX] NEQ 0
    THEN
      BEGIN
      PRE_LENGTH = .FDT[.CON_SYM[.IN, CON_INDEX], FDT_FLD_SIZE];
      %IF %BLISS(BLISS32)
      %THEN
        IF .FDT[.CON_SYM[.IN, CON_INDEX], FDT_TYPE] EQL DT_P
        THEN
          PRE_LENGTH = .PRE_LENGTH / 2 + 1;
      %FI
      END;

      ! Allocate space for the constant
      OUT_PTR = ALLOC(.PRE_LENGTH);

      ! Zero the output area
      IF CH$RCHAR(.IN_CON_PTR) EQL C_QUOTE
      THEN
        CH$FILL(.CA[CA_PAD], .PRE_LENGTH, OUT_PTR[0])
      ELSE
        CH$FILL(0, .PRE_LENGTH, OUT_PTR[0]);

      ! Set the address in CFT
      ! Set the number of bytes in output string
      CFT[.IN, CFT_CON_ADR] = OUT_PTR[0] - CFT[0, BASE_];
      CFT[.IN, CFT_CON_LEN] = .PRE_LENGTH;

      ! Check for a work file name that does not begin with a quote.
      IF ITS_A_WORK_FILE AND CH$RCHAR(.IN_CON_PTR) NEQ C_QUOTE
      THEN
        BEGIN
        CH$MOVE(.PRE_LENGTH, .IN_CON_PTR, OUT_PTR[0]);
        END
      ELSE
        ! Should indent more, but ...
```

```
.. 5101      9595  3
.. 5102      9596  3
.. 5103      9597  3
.. 5104      9598  3
.. 5105      9599  3
.. 5106      9600  3
.. 5107      9601  3
.. 5108      9602  4
.. 5109      9603  4
.. 5110      9604  4
.. 5111      9605  4
.. 5112      9606  4
.. 5113      9607  4
.. 5114      9608  4
.. 5115      9609  4
.. 5116      9610  4
.. 5117      9611  4
.. 5118      9612  5
.. 5119      9613  5
.. 5120      9614  5
.. 5121      9615  5
.. 5122      9616  5
.. 5123      9617  6
.. 5124      9618  6
.. 5125      9619  6
.. 5126      9620  5
.. 5127      9621  4
.. 5128      9622  4
.. 5129      9623  4
.. 5130      9624  4
.. 5131      9625  4
.. 5132      9626  4
.. 5133      9627  5
.. 5134      9628  4
.. 5135      9629  4
.. 5136      9630  4
.. 5137      9631  4
.. 5138      9632  3
.. 5139      9633  3
.. 5140      9634  3
.. 5141      9635  4
.. 5142      M 9636  4
.. 5143      M 9637  4
.. 5144      M 9638  4
.. 5145      M 9639  4
.. 5146      M 9640  4
.. 5147      M 9641  4
.. 5148      M 9642  4
.. 5149      M 9643  4
.. 5150      M 9644  4
.. 5151      M 9645  4
.. 5152      M 9646  4
.. 5153      M 9647  4
.. 5154      M 9648  4
.. 5155      M 9649  4
.. 5156      M 9650  4
.. 5157      M 9651  4
```

```
! What kind of constant is it?
SELECTONE CH$RCHAR_A(IN_CON_PTR) OF
SET
  [C_QUOTE]: ! It's a quoted string
  BEGIN
    ! Move string to storage area,
    ! Converting doubled quotes to single

    ! Don't include the starting quote
    IN_CON_LEN = .IN_CON_LEN - 1;

    WHILE (IN_CON_LEN = .IN_CON_LEN - 1) GTR 0 DO
      BEGIN
        IF (PRE_LENGTH = .PRE_LENGTH - 1) LSS 0 THEN EXITLOOP;
        CH$WCHAR( CH$RCHAR_A(IN_CON_PTR), OUT_PTR[0] );
        IF CH$RCHAR_A(OUT_PTR) EQL C_QUOTE
          THEN
            BEGIN
              IF CH$RCHAR_A(IN_CON_PTR) NEQ C_QUOTE THEN RETURN FAIL;
              IN_CON_LEN = .IN_CON_LEN - 1;
            END;
          END;

        ! For workfiles, we may have made the string shorter
        ! (because of doubled quotes). Set CFT_CON_LEN appropriately,
        ! since RMS may not like the pad characters at the end.

        IF ITS_A_WORK_FILE
          THEN
            CFT[.IN,CFT_CON_LEN] =
              OUT_PTR[0] = (.CFT[.IN,CFT_CON_ADR] + CFT[0,BASE_]);

        END;

  [C_PERCENT]: ! Decimal, octal, or hexadecimal constant
  BEGIN
    MACRO OCT_HEX(C_OH) =
      BEGIN
        ! From low order to high order, convert each byte
        DECR I FROM .IN_CON_LEN-1 TO 0 DO
          BEGIN
            LOCAL V: BYTE; ! The input character
            LOCAL B;
            LITERAL K = %IF C_OH EQL C_HEX %THEN 4 %ELSE 3 %FI;
            SELECTONE (V = CH$RCHAR_A(IN_CON_PTR)) OF
              SET
                [C'O' TO C'9']: V = .V - C'O';
                %IF C_OH EQL C_HEX %THEN
                [C'A' TO C'F']: V = .V - C'A' + 10;
                [C'a' TO C'f']: V = .V - C'a' + 10;
                %FI
              [OTHERWISE]: RETURN FAIL;
```



```

      TES;
      B = MINU(K, .PRE_LENGTH * %BPUNIT - K * .I);
      (.OUT_PTR)<K * .I, .B> = .V;
      END;
    END %;

! Adjust input length to reflect the percent and radix chars
IN_CON_LEN = .IN_CON_LEN - 2;

! Is it dec, oct or hex (lower_cased)?
SELECT ONE LOWER_(CH$RCHAR_A(IN_CON_PTR)) OF
  SET
    [C_DEC]: RETURN FAIL;      ! ???+++
    [C_OCT]: OCT_HEX(C_OCT);    ! Octal
    [C_HEX]: OCT_HEX(C_HEX);    ! Hexadecimal
  TES;

END;

[XC'O' TO XC'9']:
BEGIN
  EXTERNAL ROUTINE
    SOR$$CONVERT;
  LOCAL
    TYP,
    F1: VECTOR[2],
    F2: VECTOR[2];
  TYP = DT_U;                ! Unsigned binary
  F1[0] = .IN_CON_LEN;
  F1[1] = .IN_CON_PTR-1;    ! Recover from previous CH$RCHAR_A
  F2[0] = .PRE_LENGTH;
  F2[1] = OUT_PTR[0];
  IF .CON_SYMC[IN, CON_INDEX] NEQ 0
  THEN
    BEGIN
      ! Grab the destination datatype, and (for BLISS32)
      ! Grab the destination size again, since SOR$$CONVERT
      ! wants DT_P in digits, while PRE_LENGTH is in bytes.
      TYP = .FDT[CON_SYMC[IN, CON_INDEX], FDT_TYPE];
      %IF %BLISS(BLISS32) %THEN
        F2[0] = .FDT[CON_SYMC[IN, CON_INDEX], FDT_FLD_SIZE];
      %FI
      IF .FDT[CON_SYMC[IN, CON_INDEX], FDT_SCALE] NEQ 0
      THEN
        RETURN FAIL;
      END;
    IF SOR$$CONVERT(.TYP, F1[0], F2[0]) NEQ SUCCESS
    THEN
      RETURN FAIL;
    END;
  [OTHERWISE]:                ! It's something else ++++
  RETURN FAIL;
TES;
```

```

: 5158      M 9652 4
: 5159      M 9653 4
: 5160      M 9654 4
: 5161      M 9655 4
: 5162      M 9656 4
: 5163      M 9657 4
: 5164      M 9658 4
: 5165      M 9659 4
: 5166      M 9660 4
: 5167      M 9661 4
: 5168      M 9662 4
: 5169      M 9663 4
: 5170      M 9664 4
: 5171      M 9665 4
: 5172      M 9666 4
: 5173      M 9667 4
: 5174      M 9668 4
: 5175      M 9669 3
: 5176      M 9670 3
: 5177      M 9671 3
: 5178      M 9672 4
: 5179      M 9673 4
: 5180      M 9674 4
: 5181      M 9675 4
: 5182      M 9676 4
: 5183      M 9677 4
: 5184      M 9678 4
: 5185      M 9679 4
: 5186      M 9680 4
: 5187      M 9681 4
: 5188      M 9682 4
: 5189      M 9683 4
: 5190      M 9684 4
: 5191      M 9685 4
: 5192      M 9686 5
: 5193      M 9687 5
: 5194      M 9688 5
: 5195      M 9689 5
: 5196      M 9690 5
: 5197      M 9691 5
: 5198      M 9692 5
: 5199      M 9693 5
: 5200      M 9694 5
: 5201      M 9695 5
: 5202      M 9696 5
: 5203      M 9697 5
: 5204      M 9698 5
: 5205      M 9699 4
: 5206      M 9700 4
: 5207      M 9701 4
: 5208      M 9702 4
: 5209      M 9703 3
: 5210      M 9704 3
: 5211      M 9705 3
: 5212      M 9706 3
: 5213      M 9707 3
: 5214      M 9708 3
```

```
: 5215      9709  2      END;  
: 5216      9710  2  
: 5217      9711  2      RETURN SUCCESS  
: 5218      9712  2  
: 5219      9713  1      END;
```

```
                                07FC 00000  
                                24 C2 00002  
0118 CB 0128 CB D0 00005  
      57      01 CE 0000C  
      5A      0217 31 0000F  
      50      0134 CB 9E 00012 1$:  
      OC      0C BC47 7E 00017  
      58      60 D0 0001C  
      56      04 A0 9A 00020  
      04      06 A0 9A 00024  
      AE      05 A0 9A 00028  
      6E D4 0002D  
      04      AE D5 0002F  
      1C 13 00032  
      6E D6 00034  
50      04      AE      06 C5 00036  
      50      04 AC C0 0003B  
      56      04 A0 3C 0003F  
      15      60 91 00043  
      08 12 00046  
51      56      02 C7 00048  
      56      01 A1 9E 0004C  
      00000000G 00 56 DD 00050 2$:  
      08      AE 01 FB 00052  
      22      50 D0 00059  
      0C      BE 91 0005D  
      50      0E 12 00061  
56      50      0101 CB 9A 00063  
      6E      00 2C 00068  
      08      BE 0006D  
      07 11 0006F  
56      00      00 2C 00071 3$:  
      6E      08 BE 00076  
      59      06 C5 00078 4$:  
      59      08 AC C0 0007C  
      02 A9      08 AC C3 00080  
      08      56 90 00087  
      50      01 AA 9A 0008A  
      57      50 C3 0008E  
      08      10 AE D4 00092  
      50      00 ED 00095
```

```
.TITLE CC$CON  
.IDENT \T03-038\  
  
.EXTRN EC$ERR_CNTL, SASSPC_ALLOC  
.EXTRN SOR$$CONVERT  
  
.PSECT SOR$RO_CODE,NOWRT, SHR, PIC,2  
  
.ENTRY CC$CONV CONSTANTS, Save R2,R3,R4,R5,R6,R7,- : 9481  
R8,R9,RT0  
SUBL2 #36, SP  
MOVL 296(CA), 280(CA) : 9536  
MNEGL #1, IN : 9584  
BRW 29$  
MOVAB 308(CA), R10 : 9544  
MOVAQ @CON_SYM[IN], R0 : 9551  
MOVL (R0), IN CON PTR  
MOVZBL 4(R0), IN CON LEN : 9552  
MOVZBL 6(R0), PRE_LENGTH : 9553  
MOVZBL 5(R0), 4(SP) : 9557  
CLRL (SP)  
TSTL 4(SP)  
BEQL 2$  
INCL (SP)  
MULL3 #6, 4(SP), R0 : 9560  
ADDL2 FDT, R0  
MOVZWL 4(R0), PRE_LENGTH  
CMPB (R0), #21 : 9563  
BNEQ 2$  
DIVL3 #2, PRE_LENGTH, R1 : 9565  
MOVAB 1(R1), PRE_LENGTH  
PUSHL PRE_LENGTH : 9571  
CALLS #1, SASSPC_ALLOC  
MOVL R0, OUT_PTR  
CMPB @IN_CON_PTR, #34 : 9575  
BNEQ 3$  
MOVZBL 257(CA), R0 : 9577  
MOVCS #0, (SP), R0, PRE_LENGTH, @OUT_PTR  
  
BRB 4$  
MOVCS #0, (SP), #0, PRE_LENGTH, @OUT_PTR : 9579  
  
MULL3 #6, IN, R9 : 9584  
ADDL2 CFT, R9  
SUBL3 CFT, OUT_PTR, 2(R9)  
MOVB PRE_LENGTH, (R9) : 9585  
MOVZBL 1(RT0), R0 : 9589  
SUBL3 R0, IN, R0  
CLRL 16(SP)  
CMPZV #0, #8, (R10), R0
```


				10	11	1B	0009A	BLEQU	5\$		
				OC	AE	D6	0009C	INCL	16(SP)		
		22			BE	91	0009F	CMPB	@IN_CON_PTR, #34		
					08	13	000A3	BEQL	5\$		
08	BE		OC		56	28	000A5	MOVCB	PRE_LENGTH, @IN_CON_PTR, @OUT_PTR		9592
					48	11	000AB	BRB	10\$		9589
		50		OC	BE	9A	000AD	MOVZBL	@IN_CON_PTR, R0		9598
				OC	AE	D6	000B1	INCL	IN_CON_PTR		
		22			50	91	000B4	CMPB	R0, #34		9601
					3E	12	000B7	BNEQ	11\$		
					58	D7	000B9	DECL	IN_CON_LEN		9609
					58	D7	000BB	DECL	IN_CON_LEN		9611
					27	15	000BD	BLEQ	9\$		
					56	D7	000BF	DECL	PRE_LENGTH		9613
					23	19	000C1	BLSS	9\$		
		08	BE	OC	BE	90	000C3	MOVB	@IN_CON_PTR, @OUT_PTR		9614
				OC	AE	D6	000C8	INCL	IN_CON_PTR		
		50		08	BE	9A	000CB	MOVZBL	@OUT_PTR, R0		9615
				08	AE	D6	000CF	INCL	OUT_PTR		
		22			50	91	000D2	CMPB	R0, #34		
					E4	12	000D5	BNEQ	7\$		
		50		OC	BE	9A	000D7	MOVZBL	@IN_CON_PTR, R0		9618
				OC	AE	D6	000DB	INCL	IN_CON_PTR		
		22			50	91	000DE	CMPB	R0, #34		
					D6	13	000E1	BEQL	6\$		
					0151	31	000E3	BRW	32\$		
		70		10	AE	E9	000E6	BLBC	16(SP), 16\$		9627
				08	AC	C1	000EA	ADDL3	CFT, 2(R9), R0		9630
50		02			50	83	000F0	SUBB3	R0, OUT_PTR, (R9)		
69		08			63	11	000F5	BRB	16\$		9598
					50	91	000F7	CMPB	R0, #37		9634
		25			03	13	000FA	BEQL	12\$		
					00DD	31	000FC	BRW	27\$		
		58			02	C2	000FF	SUBL2	#2, IN_CON_LEN		9659
		50		OC	BE	9A	00102	MOVZBL	@IN_CON_PTR, R0		9662
				OC	AE	D6	00106	INCL	IN_CON_PTR		
		50			20	88	00109	BISB2	#32, R0		
	00000064	8F			50	D1	0010C	CMPL	R0, #100		9664
					CE	13	00113	BEQL	8\$		
	0000006F	8F			50	D1	00115	CMPL	R0, #111		9665
					3E	12	0011C	BNEQ	17\$		
54		56			03	78	0011E	ASHL	#3, PRE_LENGTH, R4		
		50			58	D0	00122	MOVL	IN_CON_LEN, I		
					30	11	00125	BRB	15\$		
		51		OC	BE	9A	00127	MOVZBL	@IN_CON_PTR, R1		
				OC	AE	D6	0012B	INCL	IN_CON_PTR		
		53			51	90	0012E	MOVB	R1, V		
		30			51	D1	00131	CMPL	R1, #48		
					AD	19	00134	BLSS	8\$		
		39			51	D1	00136	CMPL	R1, #57		
					A8	14	00139	BGTR	8\$		
		53			30	82	0013B	SUBB2	#48, V		
	52	50			03	C5	0013E	MULL3	#3, I, R2		
	51	54			52	C3	00142	SUBL3	R2, R4, R1		
		03			51	D1	00146	CMPL	R1, #3		
					03	1B	00149	BLEQU	14\$		
		51			03	D0	0014B	MOVL	#3, R1		

08	BE	51	55	53	9A	0014E	14\$:	MOVZBL	V, R5		
			52	55	F0	00151		INSV	R5, R2, B, @OUT_PTR		
			CD	50	F4	00157	15\$:	SOBGEQ	I, 13\$		
				7E	11	0015A	16\$:	BRB	26\$		9662
		00000078	8F	50	D1	0015C	17\$:	CMPL	R0, #120		9666
				75	12	00163		BNEQ	26\$		
		54	56	03	78	00165		ASHL	#3, PRE_LENGTH, R4		
			50	58	D0	00169		MOVL	IN_CON_LEN, I		
				69	11	0016C		BRB	25\$		
			51	0C	BE	9A	0016E	18\$:	MOVZBL	@IN_CON_PTR, R1	
				0C	AE	D6	00172		INCL	IN_CON_PTR	
			52	51	90	00175		MOVB	R1, V		
			30	51	D1	00178		CMPL	R1, #48		
				0A	19	0017B		BLSS	19\$		
			39	51	D1	0017D		CMPL	R1, #57		
			52	05	14	00180		BGTR	19\$		
				30	82	00182		SUBB2	#48, V		
				30	11	00185		BRB	23\$		
		00000041	8F	51	D1	00187	19\$:	CMPL	R1, #65		
				0E	19	0018E		BLSS	20\$		
		00000046	8F	51	D1	00190		CMPL	R1, #70		
				05	14	00197		BGTR	20\$		
			52	37	82	00199		SUBB2	#55, V		
				19	11	0019C		BRB	23\$		
		00000061	8F	51	D1	0019E	20\$:	CMPL	R1, #97		
				03	18	001A5		BGEQ	22\$		
		00000066	8F	008D	31	001A7	21\$:	BRW	32\$		
				51	D1	001AA	22\$:	CMPL	R1, #102		
			52	F4	14	001B1		BGTR	21\$		
			50	8F	80	001B3		ADDB2	#-87, V		
		51	50	02	78	001B7	23\$:	ASHL	#2, I, R1		
		51	54	51	C3	001BB		SUBL3	R1, R4, R1		
			04	51	D1	001BF		CMPL	R1, #4		
				03	1B	001C2		BLEQU	24\$		
			51	04	D0	001C4		MOVL	#4, R1		
			53	51	D0	001C7	24\$:	MOVL	R1, B		
		51	50	02	78	001CA		ASHL	#2, I, R1		
			55	52	9A	001CE		MOVZBL	V, R5		
08	BE	53	51	55	F0	001D1		INSV	R5, R1, B, @OUT_PTR		
			94	50	F4	001D7	25\$:	SOBGEQ	I, 18\$		
				4D	11	001DA	26\$:	BRB	29\$		9598
			30	50	91	001DC	27\$:	CMPB	R0, #48		9671
				56	1F	001DF		BLSSU	32\$		
			39	50	91	001E1		CMPB	R0, #57		
				51	1A	001E4		BGTRU	32\$		
			51	02	D0	001E6		MOVL	#2, TYP		9679
		1C	AE	58	D0	001E9		MOVL	IN_CON_LEN, F1		9680
		20	OC	01	C3	001ED		SUBL3	#1, IN_CON_PTR, F1+4		9681
		14	AE	56	D0	001F3		MOVL	PRE_LENGTH, F2		9682
			18	AE	D0	001F7		MOVL	OUT_PTR, F2+4		9683
			16	6E	E9	001FC		BLBC	(SP), 28\$		9684
		50	04	06	C5	001FF		MULL3	#6, 4(SP), R0		9692
			50	04	AC	00204		ADDL2	FDT, R0		
			51	60	9A	00208		MOVZBL	(R0), TYP		
		14	AE	04	A0	3C	0020B	MOVZWL	4(R0), F2		9694
				01	A0	95	00210	TSTB	1(R0)		9696
				22	12	00213		BNEQ	32\$		

		14	AE	9F	00215	28\$:	PUSHAB	F2			
		20	AE	9F	00218		PUSHAB	F1			9700
			51	DD	0021B		PUSHL	TYP			
	00000000G	00	03	FB	0021D		CALLS	#3, SOR\$\$CONVERT			
		01	50	D1	00224		CMPL	R0, #1			
			0E	12	00227		BNEQ	32\$			
	02	57	10	AC	F2 00229	29\$:	AOBLSS	CON_CNT, IN, 30\$			9541
			03	11	0022E		BRB	31\$			
		50	FDDF	31	00230	30\$:	BRW	1\$			
			01	D0	00233	31\$:	MOVL	#1, R0			9711
				04	00236		RET				
			50	D4	00237	32\$:	CLRL	R0			9713
				04	00239		RET				

; Routine Size: 570 bytes, Routine Base: SOR\$RO_CODE + 0000

: 5220	9714	1	
: 5221	9715	1	
: 5222	9716	1	END
: 5223	9717	0	ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
SOR\$RO_CODE	570	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SORT32.SRC]SORLIB.L32;1	409	103	25	34	00:00.1
-\$255\$DUA28:[SORT32.SRC]SRTSPC.L32;1	120	23	19	12	00:00.1
-\$255\$DUA28:[SORT32.SRC]SFKEYWRD.L32;1	1	1	100	9	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:SRTSPCBLI/OBJ=OBJ\$:SRTSPCBLI MSRC\$:SRTSPCBLI/UPDATE=(ENH\$:SRTSPCBLI)

CC\$CON
T03-038

L 11
16-Sep-1984 00:53:54

VAX-11 Bliss-32 V4.0-742

Page 177

: Size: 8135 code + 1216 data bytes
: Run Time: 03:17.9
: Elapsed Time: 11:16.4
: Lines/CPU Min: 2945
: Lexemes/CPU-Min: 30143
: Memory Used: 382 pages
: Compilation Complete

SR
VO

0366 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0367 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400
401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500
501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600
601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700
701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800
801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900
901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000